MAK TECHNOLOGIES

VR-Forces User Guide



Version 5.2

ST Engineering North America

Copyright © 2025 MAK Technologies
All rights Reserved. Printed in the United States.

Under copyright laws, no part of this document may be copied or reproduced in any form without prior written consent of MAK Technologies.

VR-Exchange™, VR-TheWorld™, VR-Vantage™, DI-Guy™, and DI-Guy Scenario™ are trademarks of MAK Technologies. MÄK Technologies®, VR-Forces®, RTIspy®, B-HAVE®, and VR-Link® are registered trademarks of MAK Technologies.

GL Studio® is a registered trademark of The DiSTI® Corporation.

Portions of this software utilize SpeedTree® technology (©2008 Interactive Data Visualization, Inc.). SpeedTree is a registered trademark of Interactive Data Visualization, Inc. All rights reserved.

RFView® is a registered trademark of Information Systems Laboratories (ISL), Inc.

SilverLining™ is a trademark of Sundog Software.

Terrain Profiles are based in part on the work of the Qwt project (http://qwt.sourceforge.net).

3ds Max® is a registered trademark of Autodesk, Inc.

All other trademarks are owned by their respective companies.

For third-party license information, visit the MAK ONE Product Support Knowledge Base (at https://www.mak.com/knowledge) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).

MAK Technologies
10 Fawcett Street, Suite 204
Cambridge, MA 02138 USA

Voice: +1 617.876.8085
Fax: +1 617.876.9208

info@mak.com

www.mak.com

Revision VRF-5.2-02-251017



Contents

PREFACE

How This Manual Is Organized 60
Documentation Set 66
Try the Help 66
Links Between PDFs 67
MAK Products 67
How to Contact Us 69
Telephone 69
E-mail 69
Internet 69
Post 69
Document Conventions 70
Mouse Button Naming Conventions 70
Third-Party Licenses 71

I. INTRODUCTION, INSTALLATION, AND STARTUP

1. Introduction to VR-Forces

1.1. Overview 76
1.1.1 Entity-Level and Aggregate-Level Simulation 78
1.1.2 Realistic Display of Vehicles and Terrain (3D View) 79
1.1.3 Create Complex Scenarios 79
1.1.4 Simulation Object Types Supported 80
1.1.5 Mission Planning 82
1.1.6 Simulation Object Tasks 82
1.1.7 Scripted Tasks and Set Data Requests 83
1.1.8 Tactical Graphics 83

1.1.9 Terrain Agility and Composability 84
1.1.10 Dynamic Terrain 84
1.1.11 Crowd Behaviors and Pattern of Life 84
1.1.12 Flexible, Intuitive Graphical User Interface 85
1.1.13 Overlays 85
1.1.14 Behaviors 85
1.1.15 Observer Attach Modes 85
1.1.16 Special Effects and Simulation Object Information Visualization 86
1.1.17 Dynamic Ocean 86
1.1.18 Lighting Effects 86
1.1.19 Accurate Vehicle Positioning 87
1.1.20 Batch Mode Operation 87
1.1.21 Remote Control 87
1.2. The VR-Forces Toolkit 87
1.2.1 Plug-in Architecture 88
1.2.2 SQL Database Logging 88
1.3. DI-Guy 89
1.4. Helpful Utilities 89
1.5. Third-Party Software and Content 90
1.5.1 SilverLining 90
1.5.2 Triton Ocean SDK 91
1.5.3 GL Studio 91
1.5.4 SpeedTree 92
1.5.5 3D Models, Terrain, and Graphical Content 92
1.5.6 OpenSceneGraph 93
1.5.7 osgEarth 93
1.6. Distributed Simulation Standards Supported 93

2. Installing VR-Forces

2.1. Installing VR-Forces 96
2.1.1 Installation and Data Packages 96
2.1.2 Installing VR-Forces on Windows 96
2.1.3 Installing VR-Forces on Linux Systems 97
2.1.4 Installing VR-Forces PDB Files for Debugging Purposes 98
2.1.5 Installing the Toolkit 99
2.1.6 The makData Package 100
2.1.7 Uninstalling from Windows 102
2.1.8 Uninstalling from Linux 102
2.2. The VR-Forces Directory Structure 102
2.3. Installing and Setting Up the MAK License Manager 104

2.3.1 Specifying the License Server 105
2.4. Installing an RTI 106
2.4.1 Installing the MAK RTI 107
2.5. Using MAK ONE Products on Virtual Machines 108
2.6. Localizing the Graphical User Interface 108
2.6.1 Translating Other Interface Files 110
2.6.2 Translating VR-Forces Scripts and Console Messages 111
2.6.3 Applying the Language Files 112

3. VR-Forces Application Concepts

3.1. The VR-Forces Program 114
3.2. GUI and Simulation Engine Concepts 114
3.2.1 How GUIs and Simulation Engines Work Together 114
3.2.2 How VR-Forces Simulation Engines Are Identified 115
3.2.3 VR-Forces Sessions 115
3.2.4 Coordinating Multiple GUIs 117
3.2.5 Working with Multiple Simulation Engines 118
3.3. Objects 119
3.3.1 The Object Parameter Database 119
3.3.2 Local Objects and Remote Objects 120
3.4. Representing and Managing Time in VR-Forces 120
3.4.1 Simulation Time 120
3.4.2 Time of Day 121
3.4.3 Exercise Clock Modes 122
3.5. Interactions 123
3.6. Advanced Terrain Navigation 123
3.7. Managing VR-Forces Settings 123
3.7.1 Synchronizing Settings Among VR-Forces Installations 126
3.7.2 Global Settings and Observer-Specific Settings 126

4. Running VR-Forces

4.1. Starting VR-Forces 129
4.1.1 Starting VR-Forces from the VR-Forces Launcher 129
4.1.2 Starting Independent VR-Forces Executables 132
4.1.3 Specifying a Session ID 133
4.1.4 VR-Forces Startup Tutorial 134
4.2. The VR-Forces Window 135
4.2.1 Opening New Windows 137
4.2.2 Printing the VR-Forces Display 138
4.2.3 Taking Screen Captures 138

4.3. Managing a GUI's Session Connection 139
4.3.1 Configuring Session Messages and Join at Startup 140
4.3.2 Joining a Session 141
4.3.3 Resigning from a Session 142
4.4. Using HLA Time Management 142
4.4.1 VR-Forces Simulation Time Versus Federation Time 143
4.4.2 Configuring Time Management for HLA Exercises 143
4.5. Opening a Terrain Database 144
4.5.1 Loading a Terrain Database at Startup 145
4.5.2 Displaying Terrain Correlation Warnings 146
4.5.3 Closing a Terrain 146
4.6. Exiting VR-Forces 146
4.6.1 Disabling the Quit Prompt 146
4.7. Configuring Simulation Connections 147
4.7.1 Opening the Simulation Connections Configuration Dialog Box 147
4.7.2 Adding a Simulation Connection 148
4.7.3 Editing a Simulation Connection 149
4.7.4 Simulation Connection Parameters 150
4.7.5 Deleting a Simulation Connection 152
4.7.6 Configuring Auto Start 153
4.7.7 Displaying Connection Information 153
4.8. Configuring Applications to Launch 154
4.8.1 Adding an Application 154
4.8.2 Managing the Applications that Will Launch 155
4.8.3 Using Application Groups 156
4.9. Managing Plug-ins 156
4.9.1 Loading Plug-ins 157
4.9.2 Adding a Plug-in 158
4.9.3 Specifying the DLLs for a Plug-in 159
4.9.4 Adding a Plug-in Configuration 160
4.9.5 Deleting a Plug-in Configuration 161
4.9.6 Deleting a Plug-in 161
4.9.7 Viewing a List of Loaded Plug-ins 162
4.10. The VR-Forces Log Files 162

5. Command-Line Options

5.1. Command-Line Options for vrfGui 164
5.2. Command-Line Options for vrfSim 178
5.3. Command-Line Options for vrfLauncher 187
5.3.1 Running in Combined Mode from the Command Line 188

5.3.2 Using the additionalargs-- Command-line Argument 188
5.4. Protocol-Independent Command-line Options 189
5.4.1 Specifying the Site ID and Application Number 189
5.4.2 Specifying the Language to Use in the GUI 189
5.4.3 Setting the Notification Level 189
5.4.4 Loading Plug-ins 190
5.5. Command-Line Options for HLA Federations 190
5.5.1 Specifying the Federation Execution 190
5.5.2 Specifying the FED File 191
5.5.3 Specifying the RPR FOM Version 191
5.5.4 Specifying a FOM Mapper 191
5.5.5 Specifying the RPR FOM Revision 191
5.5.6 Specifying FOM Mapper Initialization Data 191
5.5.7 Specifying a FED File That Is Appropriate for the FOM Mapper 192
5.5.8 Enabling Time Management 192
5.6. Command-Line Options for DIS Exercises 192
5.6.1 Specifying the Port Number 192
5.6.2 Specifying Point-to-Point or Multicast Operation 192
5.6.3 Specifying the Multicast Time-to-Live 193
5.6.4 Using Asynchronous I/O 193
5.6.5 Specifying the Exercise ID 193
5.6.6 Specifying the DIS Version 193

6. Optimizing Performance

6.1. Introduction 196
6.1.1 Optimizing Simulation Engine Performance 196
6.1.2 Optimizing GUI Performance 199
6.2. Monitoring the Simulation Engine Performance 200
6.2.1 Using the Tracy Profiler in the VR-Forces Simulation Engine 200
6.3. Monitoring Script Frame Rates 201
6.4. Displaying Performance Statistics 202
6.4.1 Displaying VR-Forces Performance Statistics 202
6.4.2 Using the Tracy Client 203
6.5. Clearing the Model Instancing Cache 204
6.6. Miscellaneous Performance Configuration Options 205
6.6.1 Limiting Use of Spot Reports 205
6.6.2 Using Asynchronous I/O for DIS Exercises 206
6.6.3 Using Incremental Compiling with Streamed Data 206
6.7. Setting the Tick Rate 206
6.7.1 Specifying the Tick Rate for Components 207

6.8. Configuring Graphics Quality 207
6.8.1 Alpha to Coverage 207
6.8.2 Using the Ambient Occlusion Effect 209
6.8.3 Configuring Ambient Occlusion 211
6.9. Balancing Visual Quality Against Network Performance 212
6.9.1 Enabling Configuration of Performance Settings 213
6.10. Trajectory Smoothing 213
6.10.1 Configuring Trajectory Smoothing 214
6.10.2 Configuring Articulated Part Smoothing 215
6.11. Tuning the Target Frame Rate 215
6.12. Coloring Draw Calls 215
6.13. Configuring VSync 216
6.13.1 Managing 3D Settings and Multiscreen Performance 217
6.14. Adaptive Tick Rate 217
6.15. Preloading Models 218
6.15.1 Command-Line Options for Preloading Models 219
6.16. DI-Guy Performance Settings 220
6.17. Specifying Terrain Categories 221
6.18. Using the Developer Debug Overlay 221
6.19. Using the MAK Remote Debugger 223

II. SCENARIOS

7. Creating and Running Scenarios

7.1. Creating a Scenario 228
7.1.1 Specifying Multiple Simulation Model Sets 233
7.1.2 Creating Simulation Model Set Configurations 235
7.1.3 Setting the Scenario Starting Date and Time 236
7.2. Importing Objects into Scenarios 238
7.2.1 Importing (Merging) Scenarios 239
7.2.2 Importing and Exporting MSDL 241
7.2.3 Importing Scenario Objects from CSV Files 243
7.2.4 Importing Airspace Control Orders 245
7.2.5 Importing Two-Line Element Sets 246
7.2.6 Loading Object Tags 246
7.3. Saving a Scenario 247
7.3.1 Saving a Previously Saved Scenario 248
7.3.2 Saving a Scenario to a New Name or a Different Format 249
7.4. Loading a Scenario 249
7.4.1 Loading a Recently Loaded Scenario 251

7.4.2 Loading a Scenario from the Command Line 251
7.4.3 Editing Scenario Properties 251
7.4.4 Editing the Scenario Description 253
7.4.5 Navigating Scenario Directories 254
7.4.6 Sample Scenarios 254
7.5. Load Balancing Scenarios 255
7.5.1 Load Balancing an Existing Scenario on Load 255
7.5.2 Load Balancing After Loading the Scenario 257
7.5.3 Loading a Load-Balanced Scenario 258
7.6. Running a Scenario 260
7.6.1 Changing the Simulation Speed 260
7.6.2 Pausing a Scenario 262
7.6.3 Rewinding a Scenario 262
7.6.4 Closing a Scenario 263
7.7. Reinitializing a Scenario 264
7.8. Checkpoints and Snapshots 264
7.8.1 Saving Checkpoints 265
7.8.2 Deleting Checkpoints 267
7.8.3 Creating Snapshots 267
7.8.4 Clearing the Snapshot List 268
7.9. Pausing a Scenario Automatically 268
7.9.1 Setting the Scenario End Time for a New Scenario 268
7.9.2 Setting the Scenario End Time for an Open Scenario 269
7.10. VR-Forces and Batch Mode 269
7.10.1 The Batch File 269
7.10.2 Creating a Batch File 270
7.10.3 Editing a Batch File 270
7.10.4 Running VR-Forces in Batch Mode 272
7.10.5 Recording Batch Scenarios 273
7.11. Sending Run and Pause Messages to Simulation Participants 273
7.12. Recording VR-Forces Simulations with the MAK Data Logger 274

8. Scenario Events

8.1. Introduction to Scenario Events 276
8.2. Creating a Scenario Event 276
8.2.1 Adding Content to a Scenario Event 279
8.2.2 Adding Linked Events 280
8.3. Adding Intelligence Objects 281
8.4. Starting a Scenario Event 283
8.4.1 Controlling Scenario Events from a Plan 284

8.5. Ending a Scenario Event 285
8.6. Enabling and Disabling Scenario Events 286
8.7. Sorting the Event List 287
8.8. Deleting a Scenario Event 287
8.9. Showing and Hiding Scenario Events 287

9. Target Detection and Combat Features

9.1. Displaying Spot-Reported Simulation Objects 290
9.1.1 Enabling or Disabling Spot Reports 290
9.1.2 Configuring the Spot Reports Viewpoint 291
9.1.3 Configuring the Spot Reports Certainty Level 294
9.1.4 Applying Spot Reports to Tactical Graphics 295
9.1.5 Displaying Entity Ground Truth in 3D Observer View 295
9.1.6 Displaying Labels for Spot Reports 296
9.1.7 Using Spot Reports in Tasks 296
9.1.8 Displaying Track Histories for Spot-Reported Objects 296
9.2. Displaying Tactical Data Link Messages 298
9.2.1 Configuring the Display of TDL Messages 299
9.2.2 Displaying Track Histories for TDL Messages 300
9.3. Managing Force Hostility Relationships 300
9.3.1 Changing a Force's Hostility in a Plan 302
9.4. Detecting Targets 302
9.4.1 Target Detection and Spot Reports 304
9.4.2 Lasing Targets 304
9.5. Engagement Zones 305
9.6. Electronic Warfare 306
9.7. Radar Coverage 306
9.7.1 Enabling the Display of Radar Coverage 308
9.7.2 Displaying Radar Coverage 308
9.7.3 Configuring Radar Coverage Displays 309
9.8. Using Sonar 310
9.8.1 Propulsion Noise and Sonar 310
9.9. Launching Countermeasures (Chaff and Flare) 311
9.10. Modeling Artillery Munitions 312

10. Indirect Fire, Ballistic Missiles, and Munition Detonations

10.1. Introduction to Indirect Fire 314
10.1.1 Creating an Indirect Fire Event 314
10.1.2 Editing Indirect Fire Events 318
10.1.3 Deleting Indirect Fire Events 318

10.1.4 Configuring Indirect Fire Event Default Values 318
10.2. Ballistic Missiles 319
10.2.1 Firing Ballistic Missiles 320
10.2.2 Editing Missile Target Events 321
10.2.3 Deleting Missile Target Events 322
10.3. Creating Munition Detonations 322

11. Setting Environment Conditions

11.1. Introduction to Environment Conditions 324
11.2. Setting Weather Conditions 324
11.2.1 Setting the Sky Cover, Wind Direction and Speed 324
11.2.2 Setting the Air Pressure and Temperature 327
11.2.3 Setting Visibility 327
11.2.4 Specifying Precipitation Type and Intensity 328
11.2.5 Setting Rain Accumulation 329
11.2.6 Setting Snow Depth and Elevation 329
11.3. Setting the Date and Time of Day 330
11.3.1 Setting the Date 331
11.3.2 Setting the Time of Day 332
11.3.3 Choosing the Illumination Model 332
11.4. Selecting an Environment Condition 333
11.4.1 Adding an Environment Condition 335
11.4.2 Editing an Environment Condition 335
11.4.3 Deleting an Environment Condition 335
11.5. Creating Local Weather Zones 336
11.5.1 Creating a Weather Zone Simulation Graphic 337
11.5.2 Editing a Weather Zone's Environment 338
11.6. Using Weather Simulation Objects 339
11.6.1 Creating a Wind Corridor 339
11.6.2 Creating an Air Turbulence Area 340
11.7. Displaying Weather Conditions at a Location 340
11.8. Marine Conditions 341
11.8.1 Enabling Marine Effects 342
11.8.2 Configuring Advanced Marine Conditions 343
11.8.3 Enabling Tidal Stream Wakes 345
11.8.4 Setting Ocean Quality 346
11.8.5 Specifying an Ocean Wave Spectrum 346
11.8.6 Configuring Breaking Waves 347
11.9. Setting the Thermocline 347
11.10. METOC Object Classes that Can Be Published 348

12. Scenario Files

12.1. Introduction 350
12.2. The Scenario File 351
12.2.1 Scenario Parameters 351
12.2.2 Specifying Pathnames for Scenario Files 355
12.2.3 Changing the Terrain Database for a Scenario 356
12.3. The Plan File 356
12.4. The Object Map File 357
12.5. Editing the Scenario Objects File 357
12.5.1 Saving Default Parameters to the Scenario Objects File 358
12.6. The Scenario Extras File 358
12.7. Temporary Scenario Directories 358

III. SIMULATION OBJECTS IN SCENARIOS

13. Introduction to Simulation Objects

13.1. Simulation Objects 362
13.2. How Simulation Objects Are Identified 362
13.2.1 UUIDs 363
13.2.2 Simulation Object Names 363
13.2.3 Echelon IDs 363
13.2.4 Object IDs 364
13.2.5 Labels 364
13.3. How Simulation Objects Are Organized 364
13.3.1 How Echelon IDs Are Assigned 365
13.4. How Simulation Objects Communicate 365
13.4.1 Custom Communication Models 366
13.5. VR-Forces Uses a Component Architecture 366
13.5.1 Sensors 366
13.5.2 Controllers 367
13.5.3 Actuators 367
13.6. Simulation Object Behaviors and Tasks 367
13.7. Entity-Level Modeling and Aggregate-Level Modeling 368

14. Creating and Placing Objects

14.1. Creating Objects 372
14.2. Selecting the Object to Create 373
14.2.1 Selecting the Object from the Create Object Palette 374
14.2.2 Selecting the Object to Create on the Create Menu 377

14.2.3 Filtering the Create Object Palette ... 377
14.2.4 Previewing Object Information ... 380
14.2.5 Draw Mode ... 381
14.3. Placing Objects ... 382
14.3.1 Placing an Object Using Default Values (Click to Create) ... 382
14.3.2 Specifying an Object's Properties Before You Create It (Click to Locate) ... 384
14.3.3 Default Placement of New Entities ... 386
14.3.4 Placing Entities Inside Buildings ... 386
14.3.5 Placing Simulation Objects on Other Simulation Objects ... 386
14.4. Specifying an Object's Altitude ... 386
14.4.1 Setting Altitude Dynamically ... 387
14.4.2 Setting Altitude in the Create Object or Edit Object Dialog Box ... 387
14.4.3 Specifying the Altitude for All of the Vertices in a Route ... 388
14.5. Specifying an Object's Heading Dynamically ... 389
14.6. Specifying an Object's Pitch Dynamically ... 389
14.7. Locking the Mouse to the Object Being Created ... 390
14.8. Simulation Object Resources ... 391
14.9. Moving Objects ... 391
14.9.1 Dragging an Object to a New Location ... 392
14.10. Editing Simulation Objects ... 393
14.10.1 Editing an Object's Name in the Object List ... 394
14.11. Copying and Pasting Objects ... 395
14.11.1 Copying Objects ... 396
14.11.2 Pasting Objects ... 396
14.11.3 Pasting Specific Entity Characteristics ... 397
14.12. Deleting Simulation Objects ... 399
14.13. Reinitializing Simulation Objects ... 399
14.14. Simulation Object Groups ... 400
14.15. Creating Random Versions of Similar Entity Types ... 400
14.16. Creating Cultural Features ... 400
14.16.1 Embedding Cultural Features in the Terrain ... 401
14.17. Creating Props ... 402
14.18. Adding an Object to the Favorites List ... 402
14.18.1 Removing an Object from the Favorites List ... 403

15. Embarked and Embedded Entities

15.1. Embarking and Disembarking Simulation Objects ... 406
15.1.1 Embarking and Disembarking Simulation Objects Instantly ... 407
15.1.2 Disembarking Simulation Objects Using the Disembark Command ... 410
15.2. Embedded Entities ... 411

15.2.1 Configuring an Entity to Support Embedded Entities 411
15.2.2 Assigning Tasks and Plans to Embedded Entities 413
15.2.3 Restoring an Entity that has Embedded Entities 414

16. Generating Traffic (Pattern of Life)

16.1. Generating Traffic 416
16.2. Spawn and Sink Traffic Generation 416
16.2.1 Creating Spawn Points 417
16.2.2 Creating Sink Points 418
16.2.3 Spawn Patterns 419
16.2.4 Spawn Pattern Management 422
16.3. Clutter Areas 428
16.3.1 Aircraft Clutter Area 428
16.3.2 Ship Clutter Area 429

17. Crowds and Herds

17.1. Crowds and Herds in Scenarios 432
17.2. Crowd and Herd Units 432
17.2.1 Creating Crowd Units 432
17.2.2 Creating Herd Units 433
17.3. Crowd and Herd Areas 434
17.3.1 Creating Crowds Using a Pedestrian Area 434
17.3.2 Creating Animal Herds Using Areas 434
17.3.3 Adding Entities to a Pedestrian or Herd Area 435
17.4. Crowd Tasks 435
17.5. Pedestrian-Oriented Tactical Graphics 435
17.5.1 Using Civilian Visit Points 436

18. Creating and Controlling Units

18.1. Creating Units 438
18.1.1 Creating a Unit by Combining Existing Simulation Objects 438
18.1.2 Creating a Preconfigured Unit 440
18.2. Selecting a Unit 440
18.3. Adding Simulation Objects to a Unit 440
18.4. Removing a Simulation Object from a Unit 441
18.5. Deleting a Unit 442

19. Displaying Units

19.1. Selecting Units 444

19.2. Expanding and Collapsing Units 444
19.2.1 Expanding or Collapsing a Single Unit 445
19.2.2 Expanding or Collapsing All Units 445
19.3. Displaying Ghosted Icons 446
19.4. Displaying Unit Icons and Bounding Volumes 448
19.4.1 Specifying the Color Scheme for Unit Bounding Volumes 450

20. Selecting Objects

20.1. The Objects List Panel 452
20.2. Selecting Simulation Objects, Tactical Graphics, and Props 453
20.2.1 Selecting Objects in the Window 455
20.2.2 Selecting Objects in the Objects List Panel 456
20.2.3 Selecting a Simulation Object in the Object Console Summary Panel 456
20.2.4 Selecting the Parent Unit of an Entity 457
20.2.5 Unselecting Objects 457
20.3. Using Selection Groups 457
20.3.1 Creating a Selection Group 458
20.3.2 Selecting a Selection Group 459
20.3.3 Adding Objects to a Selection Group 460
20.3.4 Removing Objects from Selection Groups 460
20.3.5 Renaming a Selection Group 460
20.3.6 Deleting a Selection Group 460

21. Viewing Information About Objects

21.1. Displaying Information About an Object 462
21.1.1 Information Dialog Boxes for Entity-Level Scenarios 462
21.1.2 Information Dialog Boxes for Aggregate-Level Scenarios 465
21.1.3 Information Dialog Boxes for Tactical Graphics 467
21.1.4 Viewing Information for Multiple Objects 468
21.2. Displaying Entity Labels 469
21.2.1 Turning Entity Labels On and Off 471
21.2.2 Pinning Symbol Decorations to the Window 472
21.2.3 Symbol Decorations 473
21.2.4 Configuring Shadowing and Background Color for 2D Labels 476
21.3. Using Extended Labels 477
21.3.1 Adding an Extended Label 478
21.3.2 Changing an Extended Label's Index 478
21.3.3 Deleting an Extended Label 479
21.4. Customizing 2D Icon Colors 479
21.5. Changing Force Colors 479

21.6. Displaying HAT Lines for Entities (3D Only) 480
21.7. Displaying Entity Resources 481
21.7.1 Viewing Numerical Resource Data 481
21.8. Displaying Bounding Volumes 481
21.8.1 Enabling and Disabling Bounding Volumes for Entities 482
21.8.2 Showing Bounding Volumes Only for Selected Entities 482
21.9. Configuring Object Console Messages 483
21.9.1 Setting the Notification Level for Console Messages 483
21.9.2 Viewing All Console Messages 484
21.9.3 Answering Questions from Scripted Tasks 485
21.9.4 Saving Console Messages 485
21.9.5 Displaying an Object Console Warning Icon 486
21.9.6 Clearing the Object Console 487
21.10. Viewing Object Counts 487

22. Creating and Using an Order of Battle

22.1. Introduction to the Order of Battle 490
22.1.1 The Order of Battle Panel 490
22.2. Creating an Order of Battle 491
22.2.1 Adding Objects Directly to the Order of Battle 491
22.2.2 Adding Existing Objects to the Order of Battle 493
22.3. Creating Units in the Order of Battle Panel 493
22.4. Instantiating Members of an Order of Battle in a Scenario 493
22.4.1 The Order of Battle Palette 494
22.5. Removing Objects from an Order of Battle 494
22.6. Editing the Objects in an Order of Battle 494
22.7. Changing the Order of Objects in an Order of Battle 495
22.8. Exporting an Order of Battle 495
22.9. Importing an Order of Battle 495

IV. ENTITY-LEVEL MODELING CONCEPTS

23. Ground Vehicle Movement

23.1. Overview 500
23.2. How the Move To Task Works 500
23.2.1 Path Planning 500
23.2.2 Move Execution 502
23.2.3 Dynamic Obstacle Avoidance 502
23.2.4 Vehicle Movement On Roads 503
23.3. How the Move Along Route Task Works 505

23.4. How the Follow Entity Task Works 505
23.5. How Terrain Affects Movement 505
23.5.1 Soil Effect on Movement 505
23.5.2 Slope Effects on Movement 508
23.6. Disabling Autonomous Actions 508

24. Aircraft Behaviors

24.1. Fixed-Wing Entity Behaviors 510
24.1.1 Placement of Newly Created Fixed-Wing Entities 510
24.1.2 Fly Task Behavior Is Different from Move Task Behavior 510
24.1.3 Specifying and Maintaining Altitude for Fixed-Wing Entities 510
24.1.4 Fixed-Wing Entity Movement on the Ground and in the Air 511
24.1.5 How Fixed-Wing Entities Take Off 511
24.1.6 How Fixed-Wing Entities Land 512
24.2. Rotary-Wing Entity Behaviors 513

25. Unit Behavior

25.1. How Units Are Organized 516
25.1.1 Reorganizing Units 516
25.2. How Units Move 517
25.2.1 Unit Formations 517
25.2.2 Ground Vehicle Unit Movement 518
25.3. How Units Carry Out Tasks 518

26. Suppression and Damage

26.1. Suppression 520
26.1.1 Suppressive Fire Tasks 520
26.1.2 Effects of Suppressive Fire 520
26.1.3 Configuring Suppression Effects 521
26.2. Multi-hit Actuator 522
26.2.1 Damage Points 522
26.2.2 Damage System 522

V. AGGREGATE-LEVEL MODELING CONCEPTS

27. Aggregate-Level Modeling

27.1. The Aggregate Warfare Model 528
27.1.1 Unit Combat 528
27.1.2 Unit Footprints 529

27.1.3 Unit Posture 530
27.1.4 Unit Movement 531
27.1.5 Limited Munition Attack 532
27.1.6 Contamination Areas 532
27.1.7 Unit Sensors 533
27.2. Combat Engineering Objects 533
27.2.1 Concealed Obstacles 535
27.2.2 Creating Combat Engineering Objects 535
27.2.3 How Engineering Units Create Engineering Objects 536
27.3. Logistics 537
27.3.1 Receiving Supplies 538
27.3.2 Providing Supplies 539
27.4. Electronic Warfare 539
27.4.1 Jamming Communications 539
27.4.2 Jamming Radar 540
27.4.3 Sensing Electronic Emissions 541
27.5. Air Combat 541
27.5.1 Loadouts 542
27.6. Air Missions 542
27.7. Fuel Use and Aerial Refueling 544
27.7.1 Aerial Refueling Systems 544
27.7.2 Refueling Missions 544
27.8. Air Bases and Air Missions 545
27.8.1 Preparing Missions 546
27.8.2 Landing Air Units 547
27.8.3 Repairing Aircraft 547
27.8.4 Quick Reaction Alert Status 548

28. Movement in Aggregate-Level Scenarios

28.1. Movement Systems and Terrain 550
28.2. Individual Unit Movement 550
28.2.1 Air 550
28.2.2 Land 550
28.2.3 Surface and Subsurface 551
28.3. Group Unit Movement 551
28.3.1 Air 551
28.3.2 Land 551
28.3.3 Surface and Subsurface 551

VI. TASKS

Introduction to Tasks 554

29. Assigning Tasks

29.1. Assigning Tasks to Simulation Objects 558
29.1.1 The Command Panel 558
29.1.2 Task Procedures 560
29.1.3 C++ Tasks and Scripted Tasks 560
29.1.4 Concurrent Task Execution 561
29.1.5 How Do I Know Which Simulation Objects Can Execute a Task? 562
29.1.6 Escaping the Task Assignment Process 563
29.1.7 Specifying Parameters for Tasks 563
29.1.8 Viewing Task Status 565
29.1.9 Filtering the Object Selection Lists 565
29.1.10 Skipping (Stopping) a Task 566
29.2. The Entity Control Panel 567
29.2.1 Specifying Entity Movement Using the Entity Control Panel 568
29.2.2 Using the Buttons in the Entity Control Panel 569
29.3. Using a Keyboard Shortcut to Assign a One-Click Task (or Set) 571
29.4. Assigning Tasks to Units 572
29.4.1 Convoy Tasks 572
29.4.2 Independently Tasking Unit Members 572
29.5. Reactive Tasks 572
29.5.1 Enabling Reactive Tasks 574
29.5.2 Disabling Reactive Tasks 574
29.5.3 Setting the Priority of a Reactive Task 575
29.5.4 Managing Reactive Tasks 576
29.5.5 Canceling a Reactive Task 576
29.6. Using Behavior Sets to Manage Scripts 577
29.6.1 Creating Behavior Sets 578
29.6.2 Editing Behavior Sets 579
29.6.3 Assigning a Behavior Set to a Force 580
29.7. Configuring Task Execution Rules 580
29.7.1 Configuring Action Categories 581

30. Movement Tasks for Entity-Level Scenarios

30.1. Airdrop Along Route 585
30.2. Animated Movement 585
30.3. Come to Stop 586
30.4. Convoy Along 586

30.5. Convoy To 587
30.6. Crank Left and Crank Right 587
30.7. Establish Refueling Station 587
30.8. Fly Altitude 589
30.9. Fly Cloverleaf Pattern 589
30.10. Fly Figure 8 Pattern 590
30.11. Fly Heading and Altitude 591
30.12. Fly Heading 592
30.13. Fly Loiter Pattern 592
30.14. Follow Entity 593
30.14.1 How Fixed-Wing Entities Follow Entities 594
30.15. Hover 594
30.16. Keep Station 595
30.17. Land At Airport 596
30.18. Land at Airport Near Location 596
30.19. Land On Route 596
30.20. Land On Runway 597
30.21. Land (Vertical) 598
30.22. Maneuver Along 598
30.23. Maneuver To 599
30.24. Move Along Route 599
30.25. Move Along Route with Actions 600
30.26. Move Into Formation 603
30.27. Move Next to Target 603
30.28. Move To 604
30.29. Move to Altitude 605
30.30. Move to Bullseye Offset 605
30.31. Move to Depth 606
30.32. Move To (Plan Path) 606
30.33. Orbit 607
30.34. Orbit Object 607
30.35. Paratrooper Air Drop 608
30.36. Patrol Area 609
30.37. Patrol Between 609
30.38. Patrol Route 609
30.39. Pattern Hold 610
30.40. Refuel From 610
30.41. Return To Base 611
30.42. Rotary Wing Airlift Cargo to Location 611
30.43. Rotary Wing Land 612
30.44. Rotary Wing Retrieve Cargo 612

30.45. Rotary Wing Unload Cargo 612
30.46. Sail Heading 613
30.47. Search and Rescue 613
30.47.1 Fly Creeping Line Pattern 613
30.47.2 Fly Expanding Square Pattern 615
30.47.3 Fly Parallel Pattern 616
30.47.4 Sector Search 618
30.48. Sweep Naval Mines 619
30.49. Takeoff On Route 619
30.50. Takeoff On Runway 620
30.51. Takeoff (Vertical) 621
30.52. Transition Into Formation 621
30.53. Turn to Heading 622
30.54. Zig-Zag to Heading 622

31. Engagement Tasks for Entity-Level Scenarios

31.1. Arm Mine at Depth 624
31.2. Attack Aircraft 625
31.3. Attack Aircraft with Guns 625
31.4. Cease Fire 625
31.5. Drop Naval Depth Charge 625
31.6. Drop Naval Depth Charge at Location 626
31.7. Drop Naval Mine 626
31.8. Drop Naval Mines Along Route 628
31.9. Execute Gun Attack 628
31.10. Execute Surface Attack 630
31.11. Explode Charge at Depth 631
31.12. Fire At 632
31.13. Fire Ballistic Gun 632
31.14. Fire Cruise Missile 633
31.15. Fire for Effect 634
31.15.1 Firing Naval Guns 635
31.16. Fire Javelin at Target 635
31.17. Fire Laser Guided Missile 636
31.18. Fire Missile Launcher 636
31.19. Fire Team Suppress Target 637
31.20. Lase and Fire Guided Missile 637
31.21. Lase Target 637
31.22. Launch Anti-Submarine Missile (Vertical) 638
31.23. Launch Ballistic Missile at Location 639

31.24. Launch Ballistic Missile at Target Object 639
31.25. Launch Countermeasures 640
31.26. Launch Smoke 640
31.27. Launch Torpedo Tasks 641
31.27.1 Anti-ship (Fixed Depth) 641
31.27.2 Anti-Submarine 641
31.28. Loiter Munition Tasks 642
31.28.1 Launch Loiter Munition 642
31.28.2 Loitering Attack 642
31.29. Place IED 643
31.30. Provide Indirect Fire 644
31.31. Provide Suppressive Fire 645
31.32. Release Bomb Tasks 646
31.33. Release Unguided Bomb 647
31.34. Throw Grenade Tasks 648
31.34.1 Throw Grenade at Location 648
31.34.2 Throw Grenade at Target 648

32. Human and Crowd Tasks

32.1. Circle Around Location 650
32.2. Civilian Idle 650
32.3. Civilian Pray 650
32.4. Civilian Protest 651
32.5. Civilian Talk 651
32.6. Close Door 651
32.7. Close Window 651
32.8. Create More Entities 652
32.9. Crowd Along Line 652
32.10. Crowd Around 652
32.11. Crowd Around Waypoints 653
32.12. Crowd in Front of Entity 654
32.13. DI-Guy Animation 654
32.14. DI-Guy Gesture 654
32.15. Disperse Crowd 655
32.16. Find Cover 655
32.17. Find Firing Position 656
32.18. Fire Team Bound to Location 657
32.19. Fire Team Occupy Fire Position 658
32.20. Fire Team Travel to Waypoint 658
32.21. Flee from Entities 659

32.22. Move Between Cover to Location 659
32.23. Open Door 660
32.24. Open Window 660
32.25. Parachute to Target 660
32.26. Protest Along Line 661
32.27. Protest Around Location 662
32.28. Riot 662
32.29. Rush to Location 662
32.30. Squad Travel to Waypoint 663
32.31. Visit Neighbor 663
32.32. Wait in Queue 664
32.33. Wander 664
32.34. Wander In Area 665

33. Unit Behavior Tasks for Entity-Level Scenarios

33.1. Overview of Unit Behaviors 668
33.2. Drop Off Passengers at Location 669
33.3. Pick Up Passengers at Location 669
33.4. Tasks for Mounting and Dismounting Units 669
33.4.1 Dismount Headquarters 669
33.4.2 Dismount Platoon 670
33.4.3 Dismount Squad 670
33.4.4 Dismount Team 671
33.4.5 Mount Headquarters 671
33.4.6 Mount Platoon 671
33.4.7 Mount Squad 672
33.4.8 Mount Team 672
33.5. Movement Tasks for Unit Behaviors 673
33.5.1 Bound In Lanes 673
33.5.2 Bounding Overwatch Successive Continuous 673
33.5.3 Follow Along Offset Route 673
33.6. Defensive Tasks for Unit Behaviors 674
33.6.1 Assume Coil Formation 674
33.6.2 Provide Mobile Overwatch 674
33.6.3 Provide Stationary Overwatch 675
33.7. Offensive Tasks for Unit Behaviors 675
33.7.1 Assault 675
33.7.2 Attack by Fire 676
33.7.3 Attack By Fire in Rotation 676
33.7.4 Clear 676

33.7.5 Engage From Vehicle/Soldier Firing Positions 677
33.7.6 Movement To Contact 677
33.7.7 Seize Objective 678
33.7.8 Suppressive Fire 678

34. Embarkation, Wait, Radio, and Other Tasks

34.1. Embarkation Tasks 681
34.1.1 Disembark 681
34.1.2 Disembark All 681
34.1.3 Embark 681
34.1.4 Airdrop 683
34.1.5 Airdrop from Aircraft 684
34.1.6 Rope Sling Load Pick Up 685
34.1.7 Rope Sling Load Drop Off 685
34.1.8 Fast Rope Insertion 686
34.1.9 Rope Suspension Extraction 686
34.1.10 Extract Using Rope 687
34.1.11 Rope Suspension Insertion 688
34.1.12 Ladder Extraction 688
34.1.13 Extract Using Ladder 690
34.1.14 Ladder Insertion 690
34.1.15 Insert Using Ladder 691
34.1.16 Dismount 691
34.2. Internal Transfer of Fuel 692
34.3. Transfer Supplies 692
34.4. IFF Tasks 692
34.4.1 IFF Interrogate 693
34.4.2 IFF Sweep Interrogate 694
34.5. Radio Tasks 695
34.5.1 Send Radio Set 695
34.5.2 Send Radio Task 696
34.5.3 Send Text Message 697
34.6. Wait Tasks 697
34.6.1 Wait 698
34.6.2 Wait Duration 698
34.6.3 Wait Elapsed 698
34.7. Tasks for Moving Articulated Parts 698
34.7.1 Open Car Door 698
34.7.2 Close Car Door 699
34.7.3 Open Cargo Door 699

34.7.4 Close Cargo Door 700
34.7.5 Deploy Refueling Boom 700
34.7.6 Stow Refueling Boom 700
34.7.7 Deploy Port or Starboard Refueling Drogue 701
34.7.8 Stow Port or Starboard Refueling Drogue 701
34.7.9 Extend Towed Array Decoy 702
34.7.10 Retract Towed Array Decoy 702
34.7.11 Raise Periscope 702
34.7.12 Lower Periscope 702
34.8. Embedded Entity Tasks 703
34.8.1 Deploying Embedded Entities 703
34.8.2 Recovering Deployed Entities 703
34.8.3 Sector Search Operation 703
34.8.4 Deploying an Entity by Assigning a Task 704
34.9. Deploy Sonobuoy 705
34.10. Deploy Sonobuoys Along Route 705
34.11. Sonar Dip 706
34.12. Release Next Entity From Queue 706
34.13. User Task 707
34.14. Reactive Tasks 707
34.15. Background Processes 709

35. Tasks for Aggregate-Level Scenarios

35.1. Introduction to Aggregate-Level Tasks 713
35.2. Engagement Tasks 713
35.2.1 Air-to-Air Attack 713
35.2.2 Air-to-Ground Air Base Attack 713
35.2.3 Air-to-Ground Attack 714
35.2.4 Attack by Fire 714
35.2.5 Attack with Anti-Air Missile 715
35.2.6 Attack with Anti-Ship Missile 715
35.2.7 Attack with Land-Attack Missile 715
35.2.8 Attack with Torpedo 716
35.2.9 Automatic Air Defense 716
35.2.10 Biological Attack 716
35.2.11 Bomb Location 717
35.2.12 Chemical Attack 717
35.2.13 Indirect Fire 718
35.2.14 Nuclear Attack 718
35.3. Combat Engineering Tasks 719

35.3.1 Breach Obstacles 719
35.3.2 Construct Abatis 720
35.3.3 Construct Anti-Tank Ditch 721
35.3.4 Construct Barbed Wire 721
35.3.5 Construct Barricade 722
35.3.6 Construct Berm 722
35.3.7 Construct Booby Trap 722
35.3.8 Construct Bridge 723
35.3.9 Construct Dry Gap 723
35.3.10 Construct Fortified Area 724
35.3.11 Construct Fortified Line 724
35.3.12 Construct Minefield 725
35.3.13 Construct Strong Point 725
35.3.14 Destroy Bridge 726
35.3.15 Destroy Ditch 726
35.3.16 Destroy Explosive 727
35.3.17 Destroy Fortification 727
35.3.18 Destroy Obstacle 727
35.3.19 Improve Booby Trap 727
35.3.20 Improve Breach 728
35.3.21 Improve Bridge 728
35.3.22 Improve Ditch 728
35.3.23 Improve Fortification 729
35.3.24 Improve Obstacle 729
35.4. Air Base Tasks 730
35.4.1 Launch Flight 730
35.4.2 Launch QRA Aircraft 731
35.4.3 Prepare QRA 731
35.5. Movement Tasks for Aggregate-Level Scenarios 732
35.5.1 Change MOPP Level 732
35.5.2 Change Posture 733
35.5.3 Escort Air Unit 733
35.5.4 Halt Movement 733
35.5.5 Land at Air Base 734
35.5.6 Move 734
35.5.7 Move Along Route 734
35.5.8 Move Along Route Retrograde 734
35.5.9 Move To 734
35.5.10 Move to Retrograde 735
35.5.11 Move to Location (Plan Along Roads) 735
35.5.12 Move to Waypoint (Plan Along Roads) 735

35.5.13 Patrol Route 736
35.5.14 Patrol Between 736
35.5.15 Turn To Heading 736
35.5.16 Wait for Escort 736
35.5.17 Wait Tasks 736
35.6. Embarkation Tasks 737
35.6.1 Disembark 737
35.6.2 Disembark All 737
35.6.3 Embark 737
35.7. Radio Tasks 737
35.7.1 Send Radio Set 737
35.7.2 Send Radio Task 737
35.7.3 Send Text Message 737
35.8. Other Tasks 737
35.8.1 Activate Jammer 737
35.8.2 Deactivate Jammer 737
35.8.3 Drop Fuel Tanks 738
35.8.4 FASCAM 738
35.8.5 Refuel from Tanker 738
35.8.6 Refueling Mission 739
35.8.7 Send NBC Report 739
35.8.8 Send Obstacle Report 739
35.8.9 User Task 740
35.9. Background Process List 740

36. Creating Scripted Tasks and Sets

36.1. Introduction to Scripts 745
36.1.1 Scripted Tasks 746
36.1.2 Scripted Sets 746
36.1.3 Reactive Task Scripts 746
36.1.4 Background Process Scripts 746
36.1.5 Script Metadata 747
36.1.6 The Lua Scripting Language 747
36.2. Creating a New Script 747
36.2.1 Specifying the Script ID 753
36.2.2 Specifying a Menu Icon 753
36.2.3 Specifying an Extended Name 754
36.2.4 Specifying a Short Description 754
36.2.5 Specifying a Description 754
36.2.6 Specifying the Parameters for a Script 755

36.2.7 Specifying a Menu Location 764
36.2.8 Specifying Action Categories 765
36.2.9 Configuring a Script's Availability to a Simulation Object Type 766
36.2.10 Specifying Equivalent Scripts for a Toolbar Button 769
36.2.11 Specifying the Programming Language for a Script 770
36.2.12 Configuring Scripted Tasks to Run Concurrently 771
36.2.13 Creating Scripted Sets 771
36.2.14 Creating Reactive Tasks 772
36.2.15 Creating Background Processes 774
36.3. Saving Scripts 775
36.4. Editing a Script 776
36.5. Filtering the List of Scripts 776
36.6. Organizing Scripts into Folders 777
36.6.1 Adding a Folder 777
36.6.2 Renaming a Folder 777
36.6.3 Deleting a Folder 777
36.6.4 Adding Scripts to a Folder 777
36.6.5 Removing a Script from a Folder 777
36.7. Exporting and Importing Scripts 777
36.7.1 Importing a Script Package 778
36.8. Copying a Script 779
36.9. Deleting a Script 779
36.10. Creating a System Script 779
36.10.1 Including System Scripts on the Task and Set Menus 780
36.11. Specifying a Script Editor 781
36.12. Editing Lua Files 782

37. Writing Scripts

37.1. The VR-Forces Lua Interface 784
37.1.1 Lua Classes 784
37.1.2 Lua Modules 786
37.2. Script Loading and Execution 787
37.2.1 Script Entry Points 787
37.2.2 The Script Execution Sequence 790
37.2.3 Limitations for Checkpointing Scripts 793
37.2.4 Editing Scripts While a Scenario is Running 795
37.3. Tasks and Subtasks 795
37.3.1 Monitoring the Status of Tasks and Subtasks 796
37.3.2 Stopping Tasks 797
37.3.3 Task Parameters 797

37.4. Geometry 801
37.4.1 Location3D 801
37.4.2 Vector3D 801
37.4.3 VectorOffset3D 803
37.4.4 VectorGeoc3D 803
37.5. Coding Messages for Translation 804
37.6. Error Detection 804
37.7. Interactive Scripts 805
37.8. A Basic Scripted Task 806
37.8.1 Create and Move to Waypoint Metadata 806
37.8.2 The Create and Move to Waypoint Lua Script 809
37.9. A Simple Scripted Set 810
37.10. A Simple Reactive Task 811
37.11. Background Processes 812

38. Writing Advanced Scripts with the Behavior Engine

38.1. Overview of the VR-Forces Behavior Engine 817
38.1.1 Behavior Trees 817
38.1.2 Unit-Specific Extensions to Standard Behavior Trees 817
38.1.3 Coordination of Subordinate Behaviors 818
38.1.4 Other Behavior Representation Benefits of Behavior Trees 818
38.1.5 What Behavior Trees Are Not 819
38.2. Behavior Node Definition 819
38.2.1 Common Node Attributes 820
38.2.2 Node Execution 821
38.3. Behavior Node Types 822
38.3.1 Task Node 822
38.3.2 Condition Node 822
38.3.3 Sequence Node 823
38.3.4 Loop Node 823
38.3.5 Selector Node 824
38.3.6 Parallel Node 826
38.3.7 Role Node 828
38.3.8 Sync Node 829
38.3.9 Send Node 831
38.3.10 Receive Node 831
38.3.11 Job Node 833
38.3.12 Query Node 833
38.4. Node Member Functions 834
38.5. Behavior Roles 835

38.5.1 Role Definition 835
38.5.2 Role Assignments 837
38.6. Behavior Engine API 837
38.6.1 Declaring a Behavior 837
38.6.2 Using State Variables 838
38.6.3 Getting Role Information 838
38.6.4 Variable Definitions 838
38.6.5 Script Validation 838
38.7. Behavior Integration into Lua Tasks 839
38.7.1 Script Type 839
38.7.2 Behavior Script Components 840
38.7.3 Graphical Display of Behavior 841
38.7.4 Editing with ZeroBrane Studio 843
38.8. Programming Practice for Behaviors 844
38.8.1 Node Names 844
38.8.2 Perform Single Actions in Task Nodes 845
38.8.3 Avoid Excessive Computation in Node Functions 845
38.8.4 Put Task Parameters in State Variables 845
38.8.5 Minimize Global Communication Through State 845
38.8.6 Define Nodes in Variables and Use These Variables for Children of Other Nodes 846
38.8.7 Put Static Options in doIf-Switched Nodes 850
38.8.8 Push Conditions out of Functions and into the Tree 850

39. Using Qt Designer for Script Dialog Boxes

39.1. Introduction to Using Qt Designer for Scripts 854
39.2. Selecting Qt Designer for Dialog Box Layout 854
39.3. Adding Parameters When Using Qt Designer 854
39.3.1 Specifying Choices for the Choice Parameters in Qt Designer 857
39.3.2 Specifying Categories for a Simulation Object in Qt Designer 858
39.4. Adding Design Elements 859

VII. SET DATA REQUESTS

40. Setting the State of Entity-Level Simulation Objects

40.1. Active Sonar Mode 866
40.2. Add Jamming Target 866
40.3. Autonomous Actions Enabled 866
40.4. AIS Data 867
40.5. Aiming 868

40.6. Aircraft Lighting ... 868
40.7. Altitude ... 869
40.8. Appearance ... 869
40.9. Armed ... 870
40.10. Articulated Part ... 870
40.11. Assume Unknown is Hostile ... 870
40.12. Attached ... 871
40.12.1 Detaching a Tactical Graphic ... 871
40.13. Attention Grabbing ... 871
40.14. Automatically Close Doors ... 872
40.15. Base ... 872
40.16. Callsign ... 873
40.17. Capabilities ... 873
40.18. Collision Avoidance Enabled ... 873
40.19. Concealed ... 873
40.20. Countermeasures Auto Launch ... 874
40.21. Current Speed ... 874
40.22. Damage Percentage ... 875
40.23. Destroyed ... 875
40.24. Detonation Fuse Type ... 875
40.25. DI-Guy Characteristics (Appearance, Head, Body Weight) ... 876
40.26. DI-Guy Hand Item ... 876
40.27. Disembarked ... 877
40.28. Embarked ... 877
40.29. Emergency Beacon ... 878
40.30. Emitter ... 878
40.31. Engagement Zones ... 879
40.32. Force ... 879
40.33. Formation ... 879
40.34. Freeze Fuel Levels ... 880
40.35. Freeze Movement ... 880
40.36. Freeze Munition Levels ... 881
40.37. GPS Jammer ... 881
40.38. Ground Clamp Fidelity ... 881
40.39. Heading ... 882
40.40. Hide Occupants ... 882
40.41. IFF ... 883
40.42. Invisible ... 884
40.43. Invulnerable ... 884
40.44. Jammer State ... 885
40.45. Jamming Mode ... 885

40.46. Laser Code 885
40.47. Laser Power and Wavelength 886
40.48. Loadout 886
40.49. Location 887
40.50. Look At 888
40.51. Markings 888
40.52. Name Tag 889
40.53. Navigation Enabled 889
40.54. Navigation Preferences 890
40.55. Notify Level 890
40.56. Orbit 890
40.57. Ordered Speed 891
40.58. Pitch 892
40.59. Posture (Human) 892
40.60. Preferred Trajectory Height 893
40.61. Promote From Clutter 893
40.62. Radar Mode 894
40.63. Rank Patch 894
40.64. Remote Sensor 895
40.65. Remove Jamming Target 895
40.66. Reorganize 895
40.67. Resources 896
40.68. Restore 896
40.69. Rules of Engagement 896
40.69.1 How Fire-When-Fired-Upon Works 897
40.70. Sector Blanking 897
40.71. Sector of Responsibility 898
40.72. Sensor Aim 899
40.73. Sensor Field of View Color 899
40.74. Sensor Zoom 899
40.75. Skill Level 900
40.76. Sonar Depth 900
40.77. Spot Reports 901
40.78. Stow Hand Item 902
40.79. Subsystem Enabled 902
40.80. Superior 902
40.81. Surface Vessel Lighting 903
40.82. Surrendered 903
40.83. Synchronize Laser Code 903
40.84. TACAN Transmitter 904
40.85. Tanker Lighting 904

40.86. Target 905
40.87. Target Priority List - Add 905
40.87.1 Viewing the Target Priority List 905
40.88. Target Priority List - Clear 906
40.89. Target Priority List - Update 906
40.90. Tasked by Superior 906
40.91. TDL Messaging 907
40.92. Tracer Use 907
40.93. Undetectable 907
40.94. Unit Patches 908
40.95. Urgency (Human) 908
40.96. Set Data Requests for Units 909
40.96.1 Assign Transport 909
40.96.2 Dismounted 909
40.96.3 Dismounted Headquarters Section 909
40.96.4 Lock Embarkation 910
40.96.5 Mounted 910
40.96.6 Mounted Crew 910
40.96.7 Mounted Headquarters Section 910
40.96.8 Ordered Formation 911
40.96.9 Unloaded 911

41. Setting the State of Aggregate-Level Simulation Objects

41.1. Activity 915
41.2. Ammunition 915
41.3. Ammunition Pacing/Tracking 915
41.4. Assign Aircraft 916
41.4.1 Editing Aircraft Assignments 917
41.4.2 Deleting Aircraft Assignments 917
41.5. Assume Unknown is Hostile 918
41.6. Capabilities 918
41.7. Concealed 918
41.8. Contamination 918
41.9. Current Speed 918
41.10. Destroyed 918
41.11. Disembarked 918
41.12. Embarked 918
41.13. Emitter 918
41.14. Engagement Zones 918
41.15. Engagement Results 919

41.16. Equipment Pacing/Tracking 919
41.17. Facilities 919
41.18. Food, Water, Fuel, Oil and Lubricant 920
41.19. Food, Water, Fuel, Oil and Lubricant Pacing/Tracking 921
41.20. Force 921
41.21. Has Drag Chute 921
41.22. Heading 921
41.23. Health 921
41.24. IFF 922
41.25. Loadout 922
41.26. Location 922
41.27. MOPP Level 922
41.28. Morale 923
41.29. Notify Level 923
41.30. Ordered Speed 923
41.31. Percent Complete 923
41.32. Personnel and Supplies 923
41.33. Posture (Unit) 924
41.34. Preferred Targets 924
41.35. Radar Mode 924
41.36. Reorganize 924
41.37. Resources 924
41.38. Resources Pacing/Tracking 925
41.39. Restore 925
41.40. Resupply Mode 925
41.41. Role 926
41.42. Rules of Engagement 926
41.43. Sector of Responsibility 926
41.44. Spot Reports 926
41.45. Subsystem Enabled 926
41.46. Superior 926
41.47. Supplying 926
41.48. Tasked by Superior 927
41.49. TDL Messaging 927
41.50. Weapons Pacing/Tracking 927

VIII. PLANS

Introduction to Plans 930

42. Plan Concepts

42.1. Introduction to Individual Plans and Global Plans 932
42.2. Conditional Statements 933
42.2.1 The If Statement 935
42.2.2 Triggers 935
42.2.3 While Statements 937
42.2.4 Do Until Interrupt Statement 938
42.2.5 Wait Until Statements 938
42.2.6 Conditional Tests 939
42.3. Plan Variables 943
42.4. Viewing Plans 943
42.5. Considerations for Creating Plans 944
42.5.1 Considerations for Using Triggers 944
42.5.2 Using Concurrent Tasks in Plans 945
42.5.3 Moving In Formation 945
42.5.4 Using the Tasked-By-Superior Request in a Plan 945
42.5.5 Following Simulation Objects 945
42.5.6 Using Non-VR-Forces Simulation Objects in Plans 945

43. Writing Plans

43.1. Writing Plans for Simulation Objects 948
43.1.1 Adding a Task or Set Data Request to a Plan 949
43.1.2 Adding an If Block to a Plan 950
43.1.3 Adding a While Block to a Plan 950
43.1.4 Adding a Wait Until Statement to a Plan 951
43.1.5 Adding a Trigger to a Plan 951
43.1.6 Registering Triggers 953
43.1.7 Cleaning Up Unused Triggers 954
43.1.8 Editing a Statement 955
43.1.9 Deleting Statements from a Plan 955
43.2. Printing Plans 955
43.3. Saving Changes to a Plan 955
43.4. Building Condition Statements 956
43.4.1 Building a Condition Statement with One Condition 956
43.4.2 Building a Complex Condition 957
43.4.3 Editing Condition Statements 962
43.4.4 Specifying Names or Patterns in Condition Statements 963
43.5. Using Plan Variables 965
43.6. Creating Global Plans 966
43.6.1 Controlling When Global Plans Run (Quick Launch) 968

43.6.2 Opening an Information Dialog Box for a Global Plan 969
43.7. Adding Commands to a Plan 969
43.7.1 Adding Tasks and Sets from the Commands Menu 969
43.7.2 Sending Text Messages from the Commands Menu 972
43.7.3 Sending Console Messages 972
43.7.4 Creating Objects from Within a Plan 973
43.7.5 Deleting Objects from Within a Plan 973
43.7.6 Issuing a Plan 973
43.7.7 Sending View Control Messages 975
43.7.8 Creating a Sensor View from a Plan 978
43.7.9 Sending Test Results 979
43.8. Writing Plans for Units 980
43.9. Writing a Plan for Multiple Simulation Objects 980
43.10. Writing Plans for Remote Entities 980
43.11. Copying Plans and Plan Statements 981
43.12. Restarting a Plan 981
43.13. Abandoning a Plan 982
43.14. Editing a Plan During a Simulation 982

44. Synchronization Matrix

44.1. Overview of the Synchronization Matrix 984
44.1.1 How the Synchronization Matrix Works 984
44.1.2 The Synchronization Matrix Is an Object 985
44.1.3 Units and the Synchronization Matrix 985
44.1.4 Spot Report Viewpoint in the Synchronization Matrix 985
44.2. Building a Synchronization Matrix 986
44.2.1 Opening and Creating a Sync Matrix 986
44.2.2 Adding Entities and Units 987
44.2.3 Adding and Editing Phases 987
44.2.4 Adding and Editing Plans 988
44.2.5 Copying and Pasting Plans 989
44.2.6 Continuing a Plan 989
44.2.7 Deactivating and Activating Plans 990
44.2.8 Removing Plans 990

IX. TACTICAL GRAPHICS

45. Introduction to Tactical Graphics

45.1. Overview 994
45.2. Points 996

45.3. Phase Lines 996
45.4. Routes 997
45.5. Areas 997
45.6. Obstacles 997
45.7. Displaying Tactical Graphics 998
45.7.1 Showing and Hiding Tactical Graphics 998
45.7.2 Showing and Hiding Tactical Graphics Labels 1000
45.7.3 Displaying HAT Lines for Vertices 1000
45.7.4 Displaying HAT Lines for Routes (3D Only) 1001
45.7.5 Displaying Tactical Graphics as Black or White 1001

46. Overlays

46.1. Creating and Editing Overlays 1004
46.1.1 Creating an Overlay 1005
46.1.2 Selecting Overlays 1005
46.1.3 Locking an Overlay 1005
46.1.4 Hiding the Objects on an Overlay 1006
46.1.5 Hiding Newly Created Objects 1006
46.1.6 Changing an Overlay's Name 1007
46.1.7 Removing an Object from a User-Created Overlay 1007
46.1.8 Deleting an Overlay 1007
46.2. Saving and Loading Tactical Graphics and Overlays 1007
46.2.1 Loading Tactical Graphics and Overlays 1008
46.3. Exporting Tactical Graphics as Shapefiles 1008
46.3.1 Importing Shapefiles into Overlays 1009

47. Creating and Editing Tactical Graphics and Control Objects

47.1. Creating Tactical Graphics 1012
47.1.1 How Areas Are Drawn 1012
47.1.2 Creating Freehand Lines 1012
47.1.3 Drawing Circles and Ellipses 1013
47.1.4 Drawing Arcs 1014
47.1.5 Drawing Sector Areas 1014
47.1.6 Drawing Spider Webs 1015
47.1.7 Drawing Boxes 1016
47.1.8 Creating Volumetric (Extruded) Tactical Graphics 1016
47.1.9 Adding Text to an Overlay 1017
47.1.10 Assigning Tactical Graphics to an Overlay 1017
47.1.11 Displaying a Terrain Profile When You Create a Line 1018
47.1.12 Publishing Tactical Graphics 1019

47.2. Creating a Computed Route 1019
47.3. Creating an Intel Collection Area 1020
47.4. Creating and Editing a Minefield Area 1021
47.5. Creating Ditches, Berms, and Tank Scrapes 1023
47.6. Creating Nav Edges 1023
47.7. Creating and Using Bullseyes 1025
47.7.1 Measuring Bearing and Range 1026
47.7.2 Attaching a Spider Web to a Bullseye 1028
47.8. Attaching Tactical Graphics to a Simulation Object 1028
47.8.1 Attaching a Tactical Graphic Automatically 1029
47.9. Deleting Tactical Graphics 1029
47.10. Editing Tactical Graphics 1029
47.10.1 Editing Vertices 1030
47.10.2 Resizing Boxes and Text Objects 1032
47.10.3 Rotating an Ellipse 1032
47.10.4 Changing the Direction of a Line 1033
47.10.5 Changing the Color of a Tactical Graphic 1033
47.10.6 Changing Linear and Areal Style Properties 1033
47.11. Active and Inactive Tactical Graphics 1035

48. Setting the State of Tactical Graphics

48.1. Set Data Requests for Tactical Graphics 1038
48.2. Activation 1038
48.3. Allow Crossing 1039
48.4. Arrowhead Style 1039
48.5. Color 1040
48.6. Fill Style 1040
48.7. Force 1040
48.8. Line Width 1040
48.9. Notify Level 1041
48.10. Pen Style 1041

49. Remote Graphics

49.1. Displaying Remote Graphics 1044
49.1.1 Viewing a List of Remote Graphics 1045

X. VISUAL AND AUDIO EFFECTS

50. Displaying Tactical Information and Effects

50.1. Displaying Trailing and Decal Effects (3D Only) 1051

50.1.1 Specifying Models for Trailing Effects and Decal Effects 1052
50.1.2 Displaying Persistent Tracks 1053
50.2. Displaying Track Histories 1054
50.2.1 Configuring Track History Segments and Timeout 1055
50.2.2 Displaying HAT Lines for Track Histories (3D Only) 1056
50.2.3 Coloring an Entity's Track History 1056
50.2.4 Displaying a Terrain Profile for a Track History 1057
50.3. Viewing Fire and Detonation Effects 1057
50.4. Viewing Fire and Detonation Lines 1058
50.5. Viewing Unit Combat Graphics 1059
50.6. Visualizing Tasks 1060
50.6.1 Displaying Task Visualizations for an Entity 1062
50.6.2 Pinning Task Visualizations for an Entity 1062
50.7. Displaying Sensor Contact Lines 1063
50.7.1 Enabling the Display of Sensor Contact Lines 1063
50.7.2 Specifying Which Simulation Objects Display Sensor Contact Lines 1064
50.7.3 Configuring Sensor Contact Lines 1065
50.8. Displaying Threat and Sensor Range Rings 1065
50.8.1 Enabling and Disabling the Display of Range Rings 1067
50.8.2 Pinning Range Rings for a Simulation Object 1068
50.8.3 Configuring the Display of Range Rings 1068
50.8.4 Configuring Display of Weapon System Ranges 1070
50.9. Displaying Laser Designators 1070
50.10. Displaying Supply Lines 1071
50.11. Displaying Tactical Smoke 1072
50.12. Displaying Electromagnetic Emissions 1072
50.13. Displaying Radio Communication Lines 1073
50.13.1 Enabling Radio Communications Graphics 1074
50.13.2 Configuring the Lifetime and Display Mode 1076
50.13.3 Configuring the Color of Radio Communication Lines 1076
50.14. Enabling and Disabling Wheel Rotation 1078

51. Visualizing 2D Icons and 3D Models

51.1. Simulation Object Icons 1080
51.1.1 Changing the Size of 2D Icons 1080
51.1.2 Rotating 2D Icons to a Simulation Object's Heading 1082
51.2. Exaggerating the Scale of 3D Entity Models 1082
51.3. Hiding 3D Models 1084
51.4. Displaying DI-Guys in Vehicles 1085

52. Lighting Effects

52.1. Displaying Lighting Effects 1088
52.2. Advanced Lighting Settings 1088
52.2.1 Configuring Ambient and Diffuse Lighting 1089
52.2.2 Specifying the Emissive Lighting Scale 1090
52.3. Displaying Dynamic Lighting 1090
52.3.1 Daylight Control of Lights 1092
52.4. Displaying Ocean Planar Reflections 1092
52.4.1 Configuring Ocean Planar Reflections 1093
52.5. Lens Flare 1094
52.6. Displaying Crepuscular (Sun) Rays 1095
52.7. Configuring the Skybox Cube Map 1096
52.8. Flashlight Mode 1097
52.9. Using High Dynamic Range Lighting 1097
52.10. Disabling Lighting and Clouds 1100
52.11. Displaying Shadows (3D Only) 1100
52.11.1 Enabling Shadows on Trees 1102
52.11.2 Configuring Shadows 1102

53. Marine Effects

53.1. Displaying Water Droplets and Splash Effects 1106
53.1.1 Enabling Sea Spray 1107
53.1.2 Setting the Size of Water Droplets 1108
53.1.3 Enabling and Disabling Rotor Wash 1109
53.1.4 Enabling High Quality Rotor Wash 1110
53.2. Displaying Wakes and Spray Effects 1110
53.3. Enabling Buoyancy for Surface Entities 1111

54. Using CameraFX Sensor Views

54.1. Using Sensors 1116
54.2. Enabling Sensors 1116
54.3. Configuring Sensors 1117
54.4. Adding New Sensors 1119

55. Intervisibility and Range

55.1. Intervisibility (Line-of-Sight) 1122
55.2. Types of Intervisibility 1123
55.2.1 Point-to-Point Intervisibility 1123
55.2.2 Intervisibility Fan 1123

55.2.3 Simulation Object Intervisibility 1124
55.3. Intervisibility Objects Can Be Transient or Permanent 1124
55.3.1 Creating a Transient Intervisibility Object 1125
55.3.2 Creating a Permanent Intervisibility Object 1126
55.4. Editing an Intervisibility Object 1126
55.5. Configuring Intervisibility Lines and Fans 1127
55.6. Pinning an Intervisibility Object to a Simulation Object 1128
55.7. Showing and Hiding Intervisibility Lines 1129
55.8. Displaying an Intervisibility Line's Terrain Profile 1129
55.9. Deleting Intervisibility Objects 1130
55.10. Range Lines 1130
55.10.1 Creating a Range Line 1130
55.10.2 Deleting a Range Line 1132
55.10.3 Configuring Range Lines 1132

56. Configuring Emitter Volumes

56.1. Emitter Volumes Overview 1136
56.1.1 Configuring Emitter Volume Color 1136
56.1.2 Controlling Emitter Volume Radius 1137
56.1.3 Configuring Emitter Volume Segments 1137

57. Sound Effects

57.1. Introduction to Sound Effects 1142
57.2. Enabling Sound Effects 1142
57.2.1 Configuring Sound Effects 1143
57.2.2 Decibels Configuration 1144
57.3. Associating Sound Files with Entities and Events 1145

58. MAK Radio

58.1. MAK Radio Overview 1148
58.2. Using the Radio in VR-Forces 1148
58.2.1 MAK Radio Command-Line Options 1149
58.2.2 Operating the Radio from the Dialog Box or Application 1151
58.2.3 Operating the Radio from the Radio Control Toolbar 1153
58.3. Using the Intercom or a VOX Radio 1154
58.4. Configuring the MAK Radio 1154
58.5. Audio Settings 1158
58.6. Jamming Radios 1159

XI. THE OBSERVER AND VIEWING SCENARIOS

59. The Observer and Observer Modes

59.1. Switching Between the 2D and 3D Projections 1164
59.2. Observer Modes 1164
59.3. Changing the Observer Mode 1165
59.4. Creating and Editing Observer Modes 1166
59.4.1 Creating an Observer Mode 1166
59.4.2 Deleting an Observer Mode 1167
59.4.3 Editing an Observer Mode 1167
59.4.4 Renaming an Observer Mode 1167
59.5. Adding an Observer 1167
59.5.1 Adding an Observer on the Observer Settings Page 1168
59.5.2 Adding an Observer in the Display Engine Configuration Editor 1168
59.6. Removing an Observer 1170
59.7. Introduction to XR Mode 1171
59.8. Model Sets 1172
59.8.1 Changing the Model Set 1172
59.9. Communicating Information About the Observer 1172
59.9.1 Making the Observer Visible to Other Applications 1172
59.9.2 Viewing and Attaching to Other Observers 1172
59.10. Displaying Models for Remote Observers 1173

60. Moving the Observer

60.1. Moving the Observer 1177
60.1.1 Using the Compass to View the Observer's Heading 1178
60.1.2 Setting the Compass to Degrees or MILs 1179
60.1.3 Editing the Compass's Attributes 1180
60.2. Observer Movement Functions and Keyboard Mappings 1181
60.2.1 3D Observer Movement Coordinate Systems 1182
60.2.2 2D Observer Coordinate System 1182
60.2.3 The Attached Context and Observer Movement (3D Only) 1182
60.3. Moving the Observer from the Keyboard in the 3D View 1184
60.3.1 Navigation Options 1185
60.3.2 Orbiting 1186
60.3.3 Moving the Camera 1186
60.4. Moving the Observer Using the Mouse (3D View) 1187
60.4.1 Dragging the Terrain 1187
60.4.2 Changing the Observer's Orientation with the Mouse 1189
60.4.3 Dragging the View 1189

60.4.4 Orbiting the Terrain with the Mouse 1189
60.4.5 Teleporting the Observer 1189
60.5. Moving the Observer in the 2D View 1190
60.5.1 Moving the Observer from the Keyboard in the 2D View 1190
60.5.2 Moving the Observer Using the Mouse in the 2D View 1191
60.6. Zooming the Observer 1191
60.6.1 Zooming the Observer in the 3D Projection 1191
60.6.2 Zooming the Observer in the 2D Projection 1192
60.6.3 Setting the Scale Factor for Terrain 1192
60.6.4 Setting the Magnification LOD Angle for Terrains 1193
60.6.5 Zooming In to the Ocean Over Long Distances 1194
60.6.6 Zooming to Extents 1194
60.6.7 Configuring the LOD Center in the Observer Mode 1195
60.7. Centering the Observer on a Simulation Object 1197
60.7.1 Centering on an Object from the Object Console Summary Panel 1197
60.7.2 Centering on an Object from an Information Panel 1198
60.8. Simulating Camera Jitter 1198
60.9. Forcing Orientation North 1199
60.9.1 Specifying True, Magnetic, or Grid North 1199
60.10. Enabling and Disabling View Constraints (3D Only) 1200
60.11. Controlling Navigation Speed 1200
60.11.1 Enabling or Disabling Speed Scaling 1200
60.11.2 Changing the Observer's Movement Speed 1201
60.12. Saving and Recalling Views 1202
60.12.1 Recalling a View 1204
60.12.2 Deleting Views 1204
60.12.3 Loading Views Automatically When You Load a Terrain 1204
60.12.4 Specifying the Scenario Startup View 1205
60.12.5 Saving Views to a File 1206
60.12.6 Loading Views 1206
60.13. Using the Go To Location Panel 1207
60.14. Controlling the Observer from Other Applications 1207
60.14.1 Enabling the Processing of View Control Messages 1208

61. Attaching the Observer to Objects

61.1. Attaching the Observer to a Simulation Object or Prop 1210
61.1.1 Attaching the Observer to Another Observer 1212
61.1.2 Attaching to an Articulated Part 1212
61.1.3 Attaching the Observer in Mimic Track or Tether Track Mode 1212
61.1.4 Attaching the Observer in Mimic Location Track or Tether Location Track Mode 1213

61.1.5 Filtering the Object Attachment List 1213
61.1.6 Detaching from a Simulation Object 1213
61.1.7 Hiding the Attached Model (3D View Only) 1213
61.1.8 Attaching to Props 1214
61.1.9 Improving Performance when Attaching to Ground Objects 1214
61.2. Attach Modes in the 3D View 1214
61.2.1 Absolute Mode 1215
61.2.2 Follow Mode 1215
61.2.3 Mimic Mode 1215
61.2.4 Mimic Track Mode 1216
61.2.5 Mimic Location Track Mode 1217
61.2.6 Space Follow Mode 1217
61.2.7 Tether Mode 1218
61.2.8 Tether Track Mode 1219
61.2.9 Tether Location Track Mode 1219
61.2.10 Track Mode 1219
61.3. Attach Modes in the 2D View 1220
61.4. Attaching the Observer at Startup 1221

62. Inset Views, Sensor FOVs, HUDs, and Depth of Field

62.1. Inset Views 1224
62.2. Displaying Cockpit Displays 1225
62.3. Displaying Instrument Panels 1226
62.3.1 Configuring Entities to Use Instrument Panels 1228
62.4. Displaying Sensor Views 1228
62.4.1 Displaying a Sensor Field of View Cone 1229
62.4.2 Controlling the Sensor View 1230
62.5. Using the Depth of Field Effect 1233
62.5.1 Configuring Depth of Field 1235

XII. TERRAIN

Terrain Databases Provided with VR-Forces 1238

63. Introduction to Terrain Usage in VR-Forces

63.1. Composing Terrains (and Creating MTF Files) 1242
63.2. Supported File Formats 1242
Terrain Data 1243
Elevation Data 1243
Supported Imagery Formats 1243
Supported Feature Data Formats in the GUI 1244

63.3. Feature Layers and Props 1244
63.3.1 Geometric Representation of Point Feature Data 1244
63.3.2 Feature Data Types Supported by the Simulation Engine 1245
63.4. Coordinate Systems 1245
63.4.1 The UTM Coordinate System 1245
63.4.2 Geocentric Terrain Databases 1246
63.5. Loading Supported Databases 1246
63.6. Using Large Terrain Databases 1246
63.6.1 Terrain Paging 1247
63.6.2 Terrain Streaming 1248
63.7. Using Ground Clamping 1248
63.7.1 Configuring Ground Clamping 1249

64. Displaying Terrain Information

64.1. Introduction 1252
64.2. Displaying Grid Lines 1252
64.2.1 Configuring Grid Lines 1253
64.3. Displaying Models and Terrain in Wireframe Mode 1254
64.4. Filtering Earth File Layers 1255
64.5. Displaying Feature Data (2D) 1256
64.5.1 Displaying a List of Vector Features 1258
64.5.2 Changing the Colors Used for Feature Data 1259
64.6. Adjusting Map Colors 1260
64.6.1 Adjusting Map Colors on the Display Settings Toolbar 1262
64.7. Selecting a 2D View Layer 1262
64.8. Displaying Coordinates and Soil Type for a Location 1262
64.9. The Map Scale Overlay 1263
64.10. Configuring Visible Surfaces (GDB Soil Types) 1263

65. Using Terrain Profiles

65.1. Terrain Profiles 1266
65.1.1 Displaying a Line's Terrain Profile 1267
65.1.2 Configuring Terrain Profiles 1267
65.2. Analyzing Terrain Using a Terrain Profile Line 1269

66. Generating Navigation Data

66.1. Introduction 1272
66.1.1 How Navigation Data is Generated 1272
66.1.2 Path Finding 1274
66.2. Navigation Areas 1276

66.2.1 Sectorizing Navigation Areas 1277
66.2.2 Creating a Navigation Area 1277
66.2.3 Editing a Navigation Area 1280
66.2.4 Reverting Edits to a Navigation Area 1280
66.2.5 Displaying Navigation Areas 1280
66.2.6 Removing a Navigation Area 1280
66.2.7 Displaying Information about a Navigation Area 1280
66.3. Generation of Navigation Data 1281
66.3.1 Navigation Profiles 1281
66.3.2 Generating Navigation Data for a Navigation Area 1282
66.3.3 Using Newly Generated Navigation Data 1284
66.3.4 Automatic Regeneration of Navigation Data 1284
66.4. Generating Navigation Data for Entities 1285
66.5. Importing Navigation Areas and Navigation Data 1287
66.6. Displaying Navigation Data 1287
66.6.1 Opening Navigation Lab from an Information Dialog Box 1288
66.6.2 Showing Simulation Object Movement in Navigation Lab 1289

67. Aerodromes

67.1. Overview of Aerodromes 1292
67.1.1 Checking a Terrain for Aerodromes 1292
67.2. Managed Aerodromes and Runways 1292
67.2.1 Promoting a Specific Aerodrome 1292
67.2.2 Promoting Aerodromes in a Terrain 1293
67.2.3 Viewing Aerodrome Information 1294
67.2.4 Centering on an Aerodrome 1294
67.3. Aerodrome and Runway Set Data Requests 1295
67.3.1 Aerodrome Light Conditions 1295
67.3.2 Aerodrome Undercarriage Inspection System 1295
67.3.3 Lights (Advanced) 1295
67.3.4 Runway Active 1297
67.3.5 Runway Arrestor Gear 1297
67.3.6 Runway Condition 1298
67.3.7 Runway Damaged 1298
67.3.8 Runway Light Conditions 1298

XIII. CREATING AND EDITING SIMULATION MODELS

68. The Simulation Object Editor and SMSs

68.1. Introduction to the Simulation Object Editor 1305

68.2. Starting the Simulation Object Editor 1305
68.3. Simulation Model Sets 1309
68.3.1 Including Simulation Model Sets in Other Simulation Model Sets 1310
68.3.2 Enabling and Disabling Read Only Status for SMSs 1311
68.3.3 SMS Priority for Included and Multiple SMSs 1312
68.3.4 Lua Scripts in Included SMSs 1313
68.3.5 Application-Level Files in Included SMSs 1314
68.3.6 Search Paths for Referenced Files in an SMS 1314
68.3.7 Hiding Included Objects 1314
68.3.8 Promoting a Model to the Derived SMS 1314
68.3.9 Promoting a System to the Derived SMS 1316
68.4. Visual Model Definitions 1316
68.4.1 User-Created Visual Definition Files 1317
68.5. Upgrading Old Simulation Model Sets 1318
68.5.1 The Old SMS Includes a Standard SMS 1318
68.5.2 The Old SMS Was Copied from a Standard SMS 1319
68.5.3 The Old SMS Was Completely New or Copied from base.sms 1320
68.5.4 Copying Visual Data 1320
68.5.5 Running the SMS Upgrade Tool 1321
68.5.6 Migrating an SMS to a New Release by Hand 1321
68.5.7 Migrating Pre-4.7 Weapon and Damage Systems 1322
68.5.8 Converting Pre-4.8 SMSs To Use Articulated Parts 1323
68.6. Opening a Simulation Model Set 1323
68.7. Creating New Simulation Model Sets 1323
68.7.1 Creating a New SMS that Includes an Existing SMS 1324
68.7.2 Creating a New SMS with Minimal Objects 1325
68.7.3 Creating a New SMS by Copying an SMS 1325
68.8. Importing .entity Files 1326
68.9. Applying New and Updated Variables 1326
68.10. Importing and Exporting Object Information 1326
68.10.1 Exporting Object Information 1327
68.10.2 Importing Object Information 1327
68.11. Editing the Tags List 1327
68.11.1 Adding a New Tag 1327
68.11.2 Removing a Tag 1329
68.11.3 Renaming a Tag 1329
68.11.4 Undoing Tag Changes 1329
68.11.5 Using Auto Tags 1330
68.11.6 Assigning Tags Based on Visual Model Characteristics 1330
68.11.7 Exporting Tags 1330
68.11.8 Importing Tags 1331

68.11.9 Managing Tags Using the Object Information File 1331
68.12. Managing the Forces List 1331
68.12.1 Adding a Force 1331
68.12.2 Editing a Force 1333
68.12.3 Removing a Force 1333
68.13. Copying the Object List to the Clipboard 1333

69. Editing Common Parameters

69.1. Editing Generic Object Parameters 1337
69.1.1 Selecting an Object to Edit 1337
69.1.2 Editing an Object's Tags 1337
69.1.3 Changing an Object's Object Type Enumeration 1338
69.1.4 Changing an Object's Name 1340
69.1.5 Editing an Object's Description 1341
69.1.6 Specifying Whether an Object Can Be Created 1341
69.1.7 Specifying a Default User-Created Overlay 1341
69.1.8 Specifying the Platform 1342
69.1.9 Editing a Simulation Object's Country List 1342
69.1.10 Creating Icon Lists 1343
69.1.11 Undoing (Reverting) Changes to an Object 1345
69.1.12 Reverting an Inherited Object's Parameters 1345
69.2. Editing a Simulation Object's Visual Model 1346
69.2.1 Changing the 3D Model or Colorized Model 1347
69.2.2 Changing the 2D Icon Using a Font 1348
69.2.3 Changing the 2D Icon Using an Image 1349
69.2.4 Editing a Spot Report Model 1351
69.2.5 Adding a New Visual Model or Image 1351
69.2.6 Displaying Details about the Visual Model 1352
69.2.7 Using Render Settings 1354
69.2.8 Changing a Visual Model's Type 1356
69.2.9 Specifying Visual Markings 1357
69.2.10 Importing Visual Models 1359
69.2.11 Pruning Visual Models 1359
69.2.12 Exporting Visual Models 1359
69.2.13 Reloading Visual Models in the VR-Forces GUI 1360
69.3. Creating a New Simulation Object 1360
69.3.1 Creating a New Model by Copying an Existing Model 1361
69.3.2 Inheriting a New Model from an Existing Model 1361
69.4. Deleting an Object Model 1362
69.5. Creating a Screen Capture for the Create Object Palette 1362

69.6. Creating a Simulation Object Group 1362
69.6.1 Setting the Simulation Connection when Creating an SOG 1363
69.7. Editing Tactical Graphics 1363
69.7.1 Editing a Tactical Graphic's Menu Icon 1364
69.7.2 Setting the Default Values for Tactical Graphics Properties 1364
69.7.3 Specifying Tactical Graphic Properties that Can Be Edited in the GUI 1365
69.7.4 Editing the Visual Model for a Tactical Graphic 1366
69.8. Creating a Template 1369
69.9. Editing Systems in the Simulation Object Editor 1369
69.9.1 Creating a New System 1370
69.9.2 Editing Probability Tables 1371
69.9.3 Deleting a System 1374
69.9.4 Reverting a System 1374

70. Editing Entities for Entity-Level Scenarios

70.1. Editing Simulation Object Parameters 1377
70.1.1 Editing a Simulation Object's Bounding Volume (Size) 1377
70.1.2 Displaying a Simulation Object's Bounding Volume 1378
70.1.3 Specifying Support Points 1379
70.1.4 Excluding Entities from Batch Bounding Volume and Support Point Updates 1380
70.1.5 Editing Behavioral Parameters 1380
70.1.6 Adding Object Geometry to a Simulation Object 1380
70.1.7 Configuring Cultural Features To Be Embedded in the Terrain 1381
70.1.8 Ground Clamping Cultural Features 1382
70.1.9 Specifying Entity Properties that Can Be Edited in the GUI 1383
70.1.10 Editing a Lifeform's Visual Model and Animation 1383
70.1.11 Specifying a Default Appearance 1386
70.2. Editing Sensor Signatures 1387
70.3. Editing a Model's Articulated Parts 1387
70.3.1 Adding Articulated Parts 1387
70.3.2 Editing Articulated Parts 1388
70.3.3 Animating Articulated Parts 1389
70.3.4 Displaying Muzzle Flashes 1389
70.3.5 Removing Articulated Parts 1389
70.4. Editing a Simulation Object's Systems 1390
70.4.1 Adding a System 1391
70.4.2 Selecting a Damage Model 1392
70.4.3 Selecting a Movement Model 1392
70.4.4 Editing a System's Properties 1392
70.4.5 Removing a System 1392

70.4.6 Replacing a System 1393
70.5. Creating and Editing Munitions 1393
70.5.1 Creating a Simulated Munition 1393
70.5.2 Displaying Munition Detonations 1394
70.5.3 Creating a New Ammunition Type 1394
70.6. Configuring Wakes 1395
70.7. Creating Randomized Simulation Objects 1396
70.8. Configuring Scripted Entity Movement 1399
70.8.1 Configuring Ballistic Missile Movement 1399
70.8.2 Configuring Animated Movement 1401
70.8.3 Editing Movement File Configurations 1403

71. Editing Units for Entity-Level Scenarios

71.1. Introduction to Editing Units 1406
71.2. Creating a Unit for Entity-Level Scenarios 1406
71.3. Specifying an Aggregate As Option 1406
71.4. Editing Unit Composition 1407
71.4.1 Adding Subordinates 1408
71.4.2 Removing Subordinates 1409
71.4.3 Replacing Subordinates 1409
71.4.4 Changing the Order of Subordinates 1410
71.4.5 Editing a Subordinate Unit 1410
71.4.6 Defining Functions for Subordinates 1410
71.5. Editing Formations 1411
71.5.1 Adding Formations 1412
71.5.2 Removing Formations 1413
71.5.3 Renaming Formations 1413
71.5.4 Specifying the Default Formation 1413
71.5.5 Expanding and Collapsing the Display of Formations 1413
71.5.6 Displaying Bounding Boxes 1414
71.5.7 Copying Formations 1414
71.5.8 Laying Out a Formation Automatically 1415
71.5.9 Changing a Formation's Layout by Hand 1415

72. Editing Aggregate-Level Objects

72.1. Creating Objects for Aggregate-Level Modeling 1418
72.2. Configuring Simulation Objects for Aggregate-Level Modeling 1418
72.2.1 Aggregating Units into Higher Echelon Units 1419
72.3. Configuring Ammunition, Weapons, and Equipment 1420
72.3.1 Defining Ammunition, Weapons, and Equipment Items 1422

72.4. Assemblies 1424
72.4.1 Creating Assemblies 1424
72.4.2 Copying an Assembly to Your SMS 1425
72.4.3 Editing Assemblies 1425
72.4.4 Creating an Assembly from a Unit 1426
72.4.5 Adding Assemblies to a Unit 1427
72.4.6 Removing Assemblies 1428
72.5. Rolling Up Assemblies 1428
72.5.1 Creating Roll Up Rules 1428
72.6. Adding Activities 1429
72.7. Object Parameters for Aggregate-Level Modeling 1430
72.7.1 General Parameters 1431
72.7.2 Physical Parameters 1431
72.7.3 Movement Parameters 1432
72.7.4 Sensor Parameters 1434
72.7.5 Attack Parameters 1435
72.7.6 Defense Parameters 1436
72.7.7 Combat Parameters 1437
72.7.8 Electronic Warfare (EW) Parameters 1437
72.7.9 Personnel and Equipment Parameters 1438
72.7.10 Supplies Parameters 1438
72.8. Configuring Combat Engineering Objects 1439
72.9. Editing Reader/Writer Key/Value Pairs 1440

73. Configuring Embarkation

73.1. Enabling Embarkation 1442
73.2. Adjusting the View While Configuring Embarkation 1443
73.3. Configuring Slots 1444
73.3.1 Adding Slots 1444
73.3.2 Configuring Slot Parameters 1446
73.3.3 Configuring Slot Exclusions 1449
73.3.4 Specifying Ingress and Egress Techniques 1450
73.4. Adding Ingress and Egress Points 1451
73.5. Embarkation Slot Tutorial 1453
73.5.1 Configure the Super Tanker for Embarkation 1454
73.5.2 Configure Slots for the Super Tanker 1456
73.5.3 Add Ingress Points 1459
73.5.4 Add Egress Points 1460
73.5.5 Save the New Configuration 1461
73.5.6 Test the Embarkation Configuration 1461

74. Adding DI-Guy Models to VR-Forces

74.1. Adding New DI-Guy Models to VR-Forces 1464
74.1.1 Copy DI-Guy Files to the Data Directory 1464
74.2. DI-Guy Configuration Files 1464
74.2.1 Adding a New DI-Guy Animation 1465
74.3. How VR-Forces Maps DI-Guy Models 1466
74.4. Add a New DI-Guy Entity in the SOE 1466

75. The Object Parameter Database

75.1. The Object Parameter Database 1468
75.1.1 VR-Forces Classes and Object Parameters 1469
75.1.2 Object Types 1469
75.1.3 Parameter Types 1472
75.1.4 Local and Remote Subcomponents 1473
75.2. Component Descriptors and Parameters 1477
75.2.1 Common Elements of Component Descriptors 1478
75.2.2 Configuring the Priority of Components 1481
75.3. Platform Files 1481
75.4. System Definition Files 1482
75.4.1 Creating a System Definition File 1482
75.4.2 Configuring System Connections 1483
75.5. Adding Variable Bindings to Platform and System Files 1486
75.5.1 Adding a Variable Binding 1487
75.5.2 Making Variables Editable in the Simulation Object Editor 1488
75.5.3 Adding Platform Variable Bindings to the SOE 1488
75.5.4 Adding System Definition File Variables to the SOE 1490
75.6. State Properties and Parameter Properties 1490
75.6.1 Adding State Properties and Parameter Properties 1490
75.6.2 Getting and Setting State Properties Using Lua Scripts 1493
75.6.3 Displaying State Property Values in the Information Dialog Box 1493
75.6.4 An Example of Adding State Properties 1494
75.7. Configuring Units 1501
75.8. Configuring Tactical Graphics 1501
75.9. Pathname Interpretation in Simulation Model Set Files 1502

76. Configuring Sensors

76.1. Configuring Sensors 1504
76.1.1 Adding a New Sensor to a Simulation Object 1504
76.1.2 Editing a Sensor System Definition File 1505
76.1.3 Specifying a Sensor Signature 1506

76.1.4 Specifying Sensor Detection Probabilities 1510
76.1.5 Adding Detection Types (Why Doesn't My Sensor Sense Anything?) 1512
76.1.6 Adding a New Sensor Domain 1513
76.1.7 Illumination and Time of Day 1515
76.1.8 Configuring the Effects of Rain on Radar 1516
76.2. Detection Tables 1516

77. Configuring Weapon Systems

77.1. Introduction to Weapon Systems 1520
77.1.1 The Weapon System System Definition Template File 1521
77.1.2 The Weapon System System Definition File 1523
77.2. Configuring Direct Fire Weapons 1527
77.2.1 Sensing Targets 1528
77.2.2 Configuring Ammunition 1529
77.2.3 Hit Probability Tables 1530
77.2.4 Damage Probability Tables 1531
77.3. Configuring Missile Systems 1532
77.3.1 Missile Launcher 1532
77.3.2 Missile Projectile 1533
77.3.3 Laser-Guided Missile Launcher 1533
77.3.4 Laser-Guided Missile Projectile 1534
77.4. Configuring Indirect Fire Weapons 1534
77.5. Configuring Fixed Weapons 1536
77.6. Configuring Bomb Bay Weapon Systems 1536
77.7. Configuring Weapon Systems in the SOE 1537
77.7.1 Specifying Target Types 1537
77.7.2 Configuring Hit Probability 1538
77.7.3 Configuring the Ammunition to Use 1538
77.7.4 Using the Target Configuration Assistant 1539
77.7.5 Configuring the Target Priorities 1541
77.7.6 Configuring the Target Region 1542
77.7.7 Configuring Missile Launcher Systems in the SOE 1542
77.7.8 Configuring Explosive Devices 1543

78. Configuring Damage Systems

78.1. Introduction to Damage Systems 1546
78.1.1 Calculating Damage from Munitions 1546
78.2. Configuring Damage-by-Power 1546
78.2.1 Configuring the Power of a Detonation 1547
78.2.2 Configuring Detonation Power in the SOE 1550

78.2.3 The Damage File for Damage-By-Power 1551
78.2.4 Adding a New Power Type 1553
78.3. Configuring Damage in the SOE 1555
78.3.1 Configuring Damage Power Entries 1555
78.3.2 Configuring Impact Surfaces 1555
78.3.3 Configuring the Angle of Incidence 1557
78.3.4 Specifying Effect Data 1558
78.4. Viewing Inter-Object Damage Probability Results 1558
78.5. Configuring Damage for Dynamic Terrain 1560
78.5.1 Standard Terrain Damage 1560
78.5.2 High Fidelity Terrain Damage 1561
78.6. Configuring Detonation Results 1562

79. Configuring Movement Systems

79.1. Configuring Formations 1566
79.1.1 Formation Files 1566
79.1.2 Creating a User-Defined Formation 1567
79.2. Configuring Obstruction Avoidance 1568
79.2.1 Configuring Object Types 1568
79.2.2 Configuring Vector Feature Types 1568
79.3. Configuring Embarkation 1569

80. Configuring Emitters and Radios

80.1. Configuring Emitters 1572
80.2. Configuring Radios 1574
80.2.1 Publishing Radio Transmitters 1574

81. Joystick and Keyboard Controls

81.1. Using a Joystick to Control Entities 1578
81.1.1 Switching Between Systems 1578
81.1.2 Disabling Joystick Control 1579
81.2. Using the Keyboard to Control an Entity 1579
81.3. Configuring Joysticks and Keyboard Control 1579
81.3.1 Creating a Joystick Configuration 1581
81.3.2 Mapping the Joystick Stick 1582
81.3.3 Mapping Joystick Buttons and Keyboard Keys 1583
81.3.4 Configuring “Switch Between” Options 1585
81.3.5 Using Multiple Joysticks 1588
81.3.6 Specifying a Favorite Joystick Configuration 1588
81.4. Installing a Gamepad on Linux 1589

82. Attaching Components to Remote Entities

82.1. Attaching Components to Remote Entities 1592

XIV. CONFIGURING THE GRAPHICAL USER INTERFACE

83. Display Engine and Window Management

83.1. The Display Engine 1598
83.1.1 Window Types 1598
83.2. Managing Display Engine Configurations 1598
83.2.1 Adding a Window 1598
83.2.2 Adding a Channel to a Window 1601
83.2.3 Removing a Window 1602
83.2.4 Removing a Channel 1602
83.2.5 Saving a Display Engine Configuration 1602
83.2.6 Loading a Display Engine Configuration 1602
83.3. Changing a Window's Attributes 1603
83.4. Changing a Channel's Attributes 1604
83.4.1 Setting the Clipping Planes 1605
83.4.2 Specifying the Projection Resize Policy Attribute 1607
83.4.3 Changing a Channel's Frustum (Field of View) 1609
83.4.4 Changing the Viewport 1610
83.4.5 Configuring Water Visibility 1611
83.4.6 Configuring a Mirror Channel 1612
83.4.7 Channel Keywords 1613
83.5. Configuring Multichannel Displays 1613
83.5.1 Changing the Camera's Position and Orientation Offset 1615
83.6. Stereoscopic Displays 1616
83.6.1 Configuring Anaglyphic Stereo 1617
83.6.2 Configuring Polarized Stereo 1617

84. GUI Controls, Layouts, and Behaviors

84.1. Dockable Panels and Toolbars 1622
84.1.1 Docking and Undocking a Panel 1622
84.1.2 Docking Panels as Tabs 1622
84.1.3 Docking and Undocking a Toolbar 1623
84.1.4 Docking a Toolbar Vertically 1623
84.1.5 Displaying and Hiding Panels and Toolbars 1624
84.2. Saving Window Layouts 1624
84.2.1 Creating a Window Layout 1624

84.2.2 Choosing the Window Layout 1625
84.2.3 Setting a Window Layout as the Default Layout 1626
84.2.4 Reverting a Window Layout 1627
84.2.5 Updating a Window Layout 1627
84.2.6 Renaming a Window Layout 1627
84.2.7 Deleting a Window Layout 1628
84.2.8 Restoring a Factory Layout 1628
84.3. Using Tear-Off Menus 1628
84.4. Viewing the Window in Full-Screen Mode 1629
84.4.1 Hiding the Mouse in Full Screen Mode 1629
84.5. Using Context-Sensitive Menus 1630
84.6. Common VR-Forces Buttons 1630
84.7. Using the Text Chat 1631
84.7.1 Configuring Text Chat Channels for a Scenario 1632
84.8. Adding Text to the Title Bar 1633
84.9. Specifying Display Units 1634
84.10. Disabling and Enabling Dark Mode 1637

85. Configuring Toolbars, Dialog Boxes, and Menus

85.1. Introduction 1640
85.2. Configuring Menus and Menu Commands 1640
85.2.1 Alphabetizing the Task and Set Menus 1640
85.2.2 Creating New Menus 1641
85.3. Configuring Dialog Box Pages 1641
85.4. Configuring Context-Sensitive Menus 1643
85.4.1 Editing the Task Menu 1644
85.4.2 Editing the Set Menu 1645
85.5. Configuring Toolbars 1645
85.5.1 Creating Toolbars 1646
85.5.2 Specifying an Icon for a Toolbar Item 1646
85.6. Locking Main GUI Elements 1647

86. Creating and Editing Key Mappings

86.1. Introduction to Key Maps 1650
86.2. The Key Mapping Editor 1650
86.2.1 Binary Key Mappings 1651
86.3. Editing a Key Map 1651
86.3.1 Adding a Key Mapping 1651
86.3.2 Changing a Key Mapping 1653
86.3.3 Deleting a Key Mapping 1653

86.4. Using Combined Key Mappings 1654
86.5. Filtering the Function List 1655
86.6. Creating a Key Map 1655
86.7. Deleting a Key Map 1655

XV. APPENDICES

A. Troubleshooting

A.1. Using the Simulation Engine Console 1660
A.2. Performance and Display Issues 1660
A.2.1 Features You Can Disable to Improve Performance 1660
A.3. The Fault Tolerant Heap 1661
A.4. Linux CentOS 8 Memory Issue 1661
A.5. Seamless Scenes with Multiple Channels 1661
A.6. Feature Indexing for Model Layers 1662

B. MTL Syntax

B.1. Introduction 1664
B.2. MAK Technologies Lisp (MTL) Files 1664
B.2.1 Using Environment Variables in an MTL File 1665
B.2.2 Specifying Lists of Lists 1665
B.2.3 Using Conditional Statements 1665

C. The vrfSim.mtl Configuration File

C.1. The vrfSim.mtl Configuration File Parameters 1668

D. Object Parameters

D.1. The Structure of a Simulation Object Parameter 1682
D.2. The Parameter Type Hierarchy 1682
D.2.1 Vrf Base Object Parameters 1682
D.2.2 Vrf Object Parameters 1687
D.2.3 Moving Object Parameters 1687
D.2.4 Parameter Types for Entity Platforms 1688
D.3. Object Type Enumerations 1688
D.3.1 Object Super-type 1688
D.3.2 AggregateKind 1689
D.3.3 ObjectKind 1689
D.3.4 Unit Category 1689
D.3.5 Unit Specific 1690

D.3.6 Point Object Category 1691
D.3.7 Linear Object Category 1691
D.3.8 Areal Object Category 1691
D.3.9 Interaction 1691

E. Systems and System Usage

E.1. VR-Forces Systems 1694
E.2. Entity-level systems and configured scripted tasks 1694
E.3. Aggregate-level systems and configured scripted tasks 1702

MAK PRODUCTS GLOSSARY

INDEX



Preface

This manual is written for persons who will install VR-Forces® or use the VR-Forces application. The manual assumes that you are familiar with your operating system and that you know how to work in your computer's graphical window environment.

Please see release documentation for information about changes and updates to VR-Forces since this manual was completed.

How This Manual Is Organized

This manual is organized thematically in the following major sections:

- "Section I. Introduction, Installation, and Startup" on page 73
  - "Chapter 1. Introduction to VR-Forces" on page 75 briefly describes VR-Forces and its features.
  - "Chapter 2. Installing VR-Forces" on page 95 explains how to install VR-Forces, set up license management, and localize the GUI.
  - "Chapter 3. VR-Forces Application Concepts" on page 113 provides the conceptual background for the simulation process. It will help put some of the discussion in this manual into context.
  - "Chapter 4. Running VR-Forces" on page 127 describes how to start VR-Forces, load terrain databases, and configure HLA time management.
  - "Chapter 5. Command-Line Options" on page 163 describes the command-line options for vrfGui (the GUI) and vrfSim (the simulation engine).
  - "Chapter 6. Optimizing Performance" on page 195 describes how to monitor VR-Forces's performance in the Performance Monitor Panel and how to configure VR-Forces to improve performance.
- "Section II. Scenarios" on page 225
  - "Chapter 7. Creating and Running Scenarios" on page 227 explains how to create and run scenarios, how to create batch scenarios, and how to checkpoint scenarios.
  - "Chapter 8. Scenario Events" on page 275 explains how to create and run scenario events.
  - "Chapter 9. Target Detection and Combat Features" on page 289 describes spot reports, laser targeting, entity hostility, and how entities take damage.
  - "Chapter 10. Indirect Fire, Ballistic Missiles, and Munition Detonations" on page 313 explains how to configure and call in indirect fire.
  - "Chapter 11. Setting Environment Conditions" on page 323 describes how to manipulate the date, time of day, and weather conditions.
  - "Chapter 12. Scenario Files" on page 349 describes the files used to define a scenario and how to edit them.
- "Section III. Simulation Objects in Scenarios" on page 359
  - "Chapter 13. Introduction to Simulation Objects" on page 361 contains conceptual details about simulation objects.
  - "Chapter 14. Creating and Placing Objects" on page 371 explains describes the common procedures for creating simulation objects, tactical graphics, and props.
  - "Chapter 18. Creating and Controlling Units" on page 437 explains how to create and edit units.

  - "Chapter 19. Displaying Units" on page 443 describes how to view units.
  - "Chapter 20. Selecting Objects" on page 451 describes procedures common to simulation objects, control objects, and tactical graphics.
  - "Chapter 21. Viewing Information About Objects" on page 461 explains how simulation objects are displayed, how you can change the display of simulation objects, and how you can display information about simulation objects.
  - "Chapter 15. Embarked and Embedded Entities" on page 405 explains how to embark entities. It also explains how to deploy and recover embedded entities..
  - "Chapter 16. Generating Traffic (Pattern of Life)" on page 415 explains how to use the Spawn Pattern Manager to create pattern of life traffic.
  - "Chapter 17. Crowds and Herds" on page 431 explains how to create multiple simulation objects at one time and how to create and task pedestrian crowds.
  - "Chapter 22. Creating and Using an Order of Battle" on page 489 explains how to create an Order of Battle, how to import one into a scenario, and how to create the simulation objects in the Order of Battle.
- "Section IV. Entity-Level Modeling Concepts" on page 497
  - "Chapter 23. Ground Vehicle Movement" on page 499 explains how ground vehicles move on roads and terrain.
  - "Chapter 24. Aircraft Behaviors" on page 509 explains how fixed-wing and rotary-wing aircraft move and respond to tasks.
  - "Chapter 25. Unit Behavior" on page 515 describes working with units in entity-level scenarios.
  - "Chapter 26. Suppression and Damage" on page 519 explains how entity-level scenarios model combat.
- "Section V. Aggregate-Level Modeling Concepts" on page 525
  - "Chapter 27. Aggregate-Level Modeling" on page 527 describes the aggregate-level warfare model.
  - "Chapter 28. Movement in Aggregate-Level Scenarios" on page 549 explains how simulation objects move in aggregate-level scenarios.
- "Section VI. Tasks" on page 553
  - "Chapter 29. Assigning Tasks" on page 557 provides general information about assigning tasks and creating behavior sets.
  - "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583 describes movement tasks for simulation objects in entity-level scenarios.
  - "Chapter 31. Engagement Tasks for Entity-Level Scenarios" on page 623 describes engagement tasks for simulation objects in entity-level scenarios.
  - "Chapter 32. Human and Crowd Tasks" on page 649 describes tasks for humans and crowds in entity-level scenarios.

  - "Chapter 33. Unit Behavior Tasks for Entity-Level Scenarios" on page 667 describes the tasks for units that use the behavior engine.
  - "Chapter 34. Embarkation, Wait, Radio, and Other Tasks" on page 679 describes the tasks provided with VR-Forces that apply to simulation objects in entity-level scenarios.
  - "Chapter 35. Tasks for Aggregate-Level Scenarios" on page 711 describes the tasks that apply to simulation objects in aggregate-level scenarios.
  - "Chapter 36. Creating Scripted Tasks and Sets" on page 743 introduces scripts and explains how to create them. It does not cover how to write scripts.
  - "Chapter 37. Writing Scripts" on page 783 introduces the Lua script API for VR-Forces and explains how to write scripts.
  - "Chapter 38. Writing Advanced Scripts with the Behavior Engine" on page 815 provides an overview of how to write your own tasks utilizing the behavior engine.
  - "Chapter 39. Using Qt Designer for Script Dialog Boxes" on page 853 explains how to use Qt Designer to design dialog boxes for scripted tasks and sets instead of using the automatic dialog box generation feature.
- "Section VII. Set Data Requests" on page 861
  - "Chapter 40. Setting the State of Entity-Level Simulation Objects" on page 863 describes how to change the state of simulation objects in entity-level scenarios using set data requests.
  - "Chapter 41. Setting the State of Aggregate-Level Simulation Objects" on page 913 describes how to change the state of simulation objects in aggregate-level scenarios using set data requests.
- "Section VIII. Plans" on page 929
  - "Chapter 42. Plan Concepts" on page 931 describes plans and plan components.
  - "Chapter 43. Writing Plans" on page 947 explains how to write plans for simulation objects and how to write global plans.
  - "Chapter 44. Synchronization Matrix" on page 983 explains how to use the sync matrix to write coordinating plans for multiple simulation objects.
- "Section IX. Tactical Graphics" on page 991
  - "Chapter 45. Introduction to Tactical Graphics" on page 993 is a brief introduction to tactical graphics.
  - "Chapter 46. Overlays" on page 1003 describes the creation and use of overlays.
  - "Chapter 47. Creating and Editing Tactical Graphics and Control Objects" on page 1011 describes how to create tactical graphics other than points, lines, and areas, and how to edit tactical graphics.

  - "Chapter 48. Setting the State of Tactical Graphics" on page 1037 describes set data requests that apply to tactical graphics.
  - "Chapter 49. Remote Graphics" on page 1043 explains how to enable display of graphics generated by remote applications.
- "Section X. Visual and Audio Effects" on page 1047
  - "Chapter 50. Displaying Tactical Information and Effects" on page 1049 describes how to display a variety of simulation object effects such as trailing effects, shadows, and sensor volumes.
  - "Chapter 51. Visualizing 2D Icons and 3D Models" on page 1079 explains how to change the size of models and icons and hide them.
  - "Chapter 52. Lighting Effects" on page 1087 describes the lighting effects supported by VR-Forces and how to manage them.
  - "Chapter 53. Marine Effects" on page 1105 describes water related effects, such as displaying rain splash and water spray. It also covers wake effects and entity buoyancy.
  - "Chapter 54. Using CameraFX Sensor Views" on page 1115 explains how to enable and configure infrared and night vision goggle sensor views.
  - "Chapter 55. Intervisibility and Range" on page 1121 explains how to determine if simulation objects can see other simulation objects or areas of the terrain.
  - "Chapter 56. Configuring Emitter Volumes" on page 1135 explains how to configure the emitter volume displays.
  - "Chapter 57. Sound Effects" on page 1141 explains how to enable and configure sound effects for entities.
  - "Chapter 58. MAK Radio" on page 1147 explains how to use the radio features.
- "Section XI. The Observer and Viewing Scenarios" on page 1161
  - "Chapter 59. The Observer and Observer Modes" on page 1163, explains what an observer is and the different ways observers can be configured.
  - "Chapter 60. Moving the Observer" on page 1175 explains how to navigate the 3D and 2D views by moving the observer.
  - "Chapter 61. Attaching the Observer to Objects" on page 1209 explains how to attach the observer to simulation objects. This is an alternative to manually moving the observer through the terrain.
  - "Chapter 62. Inset Views, Sensor FOVs, HUDs, and Depth of Field" on page 1223 describes special secondary windows – inset views, sensor fields of view, and instrument panels, and HUDs.

- "Section XII. Terrain" on page 1237
  - "Chapter 63. Introduction to Terrain Usage in VR-Forces" on page 1241 provides conceptual background for terrain use in VR-Forces.
  - "Chapter 64. Displaying Terrain Information" on page 1251 describes ways to change how terrain is displayed. These options provide different types of information about the terrain.
  - "Chapter 65. Using Terrain Profiles" on page 1265 explains how to display and configure terrain profiles.
  - "Chapter 66. Generating Navigation Data" on page 1271 explains how to generate navigation data for advanced navigation by humans and ground entities.
  - "Chapter 67. Aerodromes" on page 1291 explains how to control airport settings either in a plan or by using the Airport Settings Panel.
- "Section XIII. Creating and Editing Simulation Models" on page 1301
  - "Chapter 68. The Simulation Object Editor and SMSs" on page 1303 explains simulation model sets (SMSs) and how to create them in the Simulation Object Editor.
  - "Chapter 69. Editing Common Parameters" on page 1335 explains how to use the Simulation Object Editor to edit simulation object parameters that are common to entity-level and aggregate-level SMSs.
  - "Chapter 70. Editing Entities for Entity-Level Scenarios" on page 1375 explains how to use the Simulation Object Editor to edit simulation object parameters, systems, and models that apply to individual entities.
  - "Chapter 71. Editing Units for Entity-Level Scenarios" on page 1405, explains how to configure unit composition and formations in the Simulation Object Editor. This chapter applies to units in entity-level scenarios.
  - "Chapter 72. Editing Aggregate-Level Objects" on page 1417 explains how to configure aggregates for aggregate-level scenarios in the Simulation Object Editor.
  - "Chapter 73. Configuring Embarkation" on page 1441 explains how to enable embarkation and configure slots, ingress points, and egress points.
  - "Chapter 74. Adding DI-Guy Models to VR-Forces" on page 1463 explains how to configure new DI-Guy characters.
  - "Chapter 75. The Object Parameter Database" on page 1467 describes the object parameter database, which contains parameters for simulation objects and control objects, and how to edit it.
  - "Chapter 76. Configuring Sensors" on page 1503 explains how to configure sensors and sensor signatures.
  - "Chapter 77. Configuring Weapon Systems" on page 1519 explains how to configure weapon systems by editing configuration files and in the Simulation Object Editor.

  - "Chapter 78. Configuring Damage Systems" on page 1545 explains how to configure damage systems by editing configuration files and in the Simulation Object Editor.
  - "Chapter 79. Configuring Movement Systems" on page 1565 explains how to configure obstacle avoidance, embarkation, and formations.
  - "Chapter 80. Configuring Emitters and Radios" on page 1571 explains how to configure the simulation of emitters and radios.
  - "Chapter 81. Joystick and Keyboard Controls" on page 1577 explains how to configure joysticks and keyboard keys to control entity movement.
  - "Chapter 82. Attaching Components to Remote Entities" on page 1591, explains how to configure sensors and sensor signatures.
- "Section XIV. Configuring the Graphical User Interface" on page 1595
  - "Chapter 83. Display Engine and Window Management" on page 1597 explains how to configure windows and channels and how to save and load display configurations.
  - "Chapter 84. GUI Controls, Layouts, and Behaviors" on page 1621 explains how to manage toolbars and other aspects of the VR-Forces GUI.
  - "Chapter 85. Configuring Toolbars, Dialog Boxes, and Menus" on page 1639 explains how to configure the menus, toolbars, and dialog box pages that get displayed.
  - "Chapter 86. Creating and Editing Key Mappings" on page 1649 explains how to edit key mappings and create new key mappings.
- "Section XV. Appendices" on page 1657.
  - "Appendix A. Troubleshooting" on page 1659 explains how to resolve some common problems that can arise with VR-Forces.
  - "Appendix B. MTL Syntax" on page 1663 explains the syntax of configuration files provided with VR-Forces.
  - "Appendix C. The vrfSim.mtl Configuration File" on page 1667 describes the parameters in *vrfSim.mtl*.
  - "Appendix D. Object Parameters" on page 1681 describes the high level parameters used in Simulation Model Sets.
  - "Appendix E. Systems and System Usage" on page 1693 lists all the systems provided with VR-Forces and provides information about them, including which entities use them.

"MAK Products Glossary" on page 1707 defines terms common to DIS, HLA, and MAK products.

The VR-Forces Quick Reference Card included with the product documentation lists command-line options, keyboard shortcuts, and navigation and view options.

Documentation Set

VR-Forces documentation is provided as manuals in PDF format, help, and HTML class documentation. Electronic versions of the documentation are in vrforcesx.x/doc. On Windows, you can access the documentation from the VR-Forces 5.2 Documentation folder on the Start menu. You can also find the most current documentation for each product's release at https://www.mak.com/mak-one/support/help/specs. The VR-Forces documentation set includes:

- *VR-Forces User Guide* contains all documentation for running VR-Forces, creating and running scenarios, and configuring VR-Forces.
- *VR-Forces Migration Guide* collates the most substantial migration details for recent releases.
- *MAK ONE Model Catalog*. A catalog of the simulation objects and tactical graphics configured in the Simulation Object Editor, with the name, object type, and a description and image 3D models where applicable.
- *Adding Content to MAK ONE Applications Reference Manual* explains how to add and edit terrain and visual models for VR-Forces.
- *MAK ONE Products Interoperability Guide* provides an overview of using MAK ONE products in DIS and HLA exercises. It explains how to configure MAK ONE products to interoperate with each other as well as other applications that adhere to the DIS or HLA standards.
- *VR-Forces First Experience Guide*. A brief introduction to the most basic features of VR-Forces.
- Help. The help, accessible from the Help menu, replicates the information in the *VR-Forces User Guide* and *Adding Content to MAK ONE Applications Reference Manual*.
- *VR-Forces Developers Guide* and API documentation. Class documentation and developers guides in linked HTML pages.
- *VR-Forces Release Notes* lists system requirements, release-specific requirements, new features and updates, bug fixes, and known problems.

Try the Help

MAK's HTML5 help is the best way to use our documentation. It's easy to navigate, with improved natural language search tools and filters to focus your search. You can click the illustrations to see high-resolution enlargements that show more detail. It's built into the product and installed locally — simply press **F1** or choose Help from the Help menu to launch the help in your browser. You do not need to be connected to the internet to use it.



If you attempt to view the help in a browser that does not fully support HTML5, you may experience issues with some features.

Links Between PDFs

Many MAK PDF documents link to each other. For example, the description of a new feature in the release notes might include links to detailed information in a user guide, and there are also many links from user guides to the *Adding Content to MAK ONE Applications Reference Manual* and back. These links between PDFs rely on two things:

- The linked PDF files must be in the same directory. MAK installers put the PDFs for each product in that product's *doc* directory — in Windows, *./vrforces/doc*. If you move or copy a PDF away from that directory or download a new one to another location, any links between that PDF and the others stop working.
- You must use Acrobat Reader to read MAK PDFs. Links between PDFs do not work when you read with a web browser. Acrobat Reader is free; you can get it from Adobe at https://get.adobe.com/reader.

MAK Products

The MAK Technologies product line includes:

- VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the NETN FOM and RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications.

  VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.
- MAK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MAK RTI is optimized for high performance. It has an API, RTIspy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you 2D and 3D views of a simulated environment.

  You can create and view entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. You can simulate using entity-level modeling, which focuses on the actions of individual people and vehicles, and aggregate-level modeling, which focuses on the interaction of large hierarchical units.

  VR-Forces also functions as a plan view display for viewing remote simulation objects taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.

- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes VR-Vantage and the VR-Vantage Toolkit.
  - VR-Vantage displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to simulation objects so that it moves as they do.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics-based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™. VR-TheWorld is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- RFView® for MAK ONE. Synthetic Aperture Radar (SAR) imaging is an indispensable tool for today's modern militaries. Developed in partnership with ISL, RFView for MAK ONE acts as a SAR server and delivers simulated SAR images that are both physically accurate and fully correlated with out-the-window views and EO/IR images of MAK ONE synthetic environments. It includes a toolkit that enables you to build a client application to request SAR images from the server.

- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.
- MAK Legion. MAK Legion is a collection of applications and an SDK designed to support simulations with millions of simulation objects.

How to Contact Us

For VR-Forces technical support, information about upgrades, and information about other MAK products, you can contact us in the following ways:

Telephone

Call or fax us at:	Voice:	617-876-8085
	Fax:	617-876-9208

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com

Internet

MAK website home page:	www.mak.com
License key requests:	https://www.mak.com/mak-one/support/portal
Technical support:	https://www.mak.com/mak-one/support/portal
Product version and platform information:	https://www.mak.com/mak-one-specs
For the free, unlicensed MAK RTI:	https://www.mak.com/mak-one/support/help/bonus-material

Post

Send postal correspondence to:	MAK Technologies 10 Fawcett Street, Suite 204 Cambridge, MA 02138 USA

When requesting support, tell us the product you are using, the version, and the platform on which you are running. If you are using VR-Vantage, VR-Forces, VR-Engage, or DI-Guy, please also include your data compatibility version and date.

Document Conventions

This manual uses these typographic conventions:

Sans Serif Bold	Indicates a key on the keyboard or menu options. Mentions of letter keys do not require you to press the **Shift** key. If Shift is necessary, that will be included, for example, **Shift**+**Q**. Also indicates parameters.
`Monospaced`	Indicates values you enter such as commands or arguments.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
{}	Indicates required arguments.
[]	Indicates optional arguments.
|	Separates options in a command where only one option may be chosen at a time.
`(|)`	In command syntax, indicates equivalent alternatives for a command-line option, for example, `(--help | -h)`.
/	Indicates a directory. Since MAK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames.
Italic	Indicates a file name, pathname, or a class name.
➤	Indicates a procedure.
Menu > Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File > Save**.
	Click the icon to run a tutorial video in your default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.
	Indicates that a section is valid only for entity-level scenarios.
	Indicates that a section is valid only for aggregate-level scenarios.

Directory names are preceded with dot and slash characters that show their position with respect to the VR-Forces home directory. For example, the directory *./vrforces5.2/doc* appears in the text as *./doc*.

For anything in the data directory, the path is shown starting with *./SharedData/19/latest* to be clear that it is not in the VR-Forces home directory. That is the default installation path. If you installed the data in a different directory, substitute your path.

Mouse Button Naming Conventions

An instruction to click the mouse button refers to clicking the primary mouse button, usually the left button for right-handed mice and the right button for left-handed mice. The context-sensitive menu, also called a popup menu or right-click menu, refers to the menu displayed when you click the secondary mouse button, usually the right button on right-handed mice and the left button on left-handed mice.

Third-Party Licenses

MAK software products may use code from third parties. For an up-to-date list of supported third-party and open source libraries and the associated license information, please visit the MAK ONE Product Support Knowledge Base (at https://www.mak.com/knowledge) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).



I. Introduction, Installation, and Startup

VR-Forces is a computer generated forces (CGF) application and toolkit. VR-Forces is implemented using separate GUI and simulation engine applications that allow tremendous flexibility for managing your simulation needs.



1. Introduction to VR-Forces

This chapter provides a high-level description of VR-Forces features.

1.1. Overview 76
1.1.1 Entity-Level and Aggregate-Level Simulation 78
1.1.2 Realistic Display of Vehicles and Terrain (3D View) 79
1.1.3 Create Complex Scenarios 79
1.1.4 Simulation Object Types Supported 80
1.1.5 Mission Planning 82
1.1.6 Simulation Object Tasks 82
1.1.7 Scripted Tasks and Set Data Requests 83
1.1.8 Tactical Graphics 83
1.1.9 Terrain Agility and Composability 84
1.1.10 Dynamic Terrain 84
1.1.11 Crowd Behaviors and Pattern of Life 84
1.1.12 Flexible, Intuitive Graphical User Interface 85
1.1.13 Overlays 85
1.1.14 Behaviors 85
1.1.15 Observer Attach Modes 85
1.1.16 Special Effects and Simulation Object Information Visualization 86
1.1.17 Dynamic Ocean 86
1.1.18 Lighting Effects 86
1.1.19 Accurate Vehicle Positioning 87
1.1.20 Batch Mode Operation 87
1.1.21 Remote Control 87
1.2. The VR-Forces Toolkit 87
1.2.1 Plug-in Architecture 88
1.2.2 SQL Database Logging 88
1.3. DI-Guy 89
1.4. Helpful Utilities 89

1.5. Third-Party Software and Content 90
1.5.1 SilverLining 90
1.5.2 Triton Ocean SDK 91
1.5.3 GL Studio 91
1.5.4 SpeedTree 92
1.5.5 3D Models, Terrain, and Graphical Content 92
1.5.6 OpenSceneGraph 93
1.5.7 osgEarth 93
1.6. Distributed Simulation Standards Supported 93

1.1. Overview

VR-Forces is a computer generated forces (CGF) application and toolkit. VR-Forces is implemented using separate applications—a graphical user interface (GUI) and a simulation engine (sim engine)—that allow tremendous flexibility for managing your simulation needs. (For information about how this works, see "3.1. The VR-Forces Program" on page 114.)

VR-Forces provides 3D and 2D views of your simulated world, integrated into one graphical user interface (GUI). The 2D view provides you with insights into a simulated battle by overlaying simulation objects and information onto 2D views of tactical, strategic, and visual databases. The 2D view answers typical questions about the placement of forces and how the terrain might affect the engagement more easily than do three-dimensional displays. It is ideally suited for viewing simulations over large areas of open terrain.

The 3D view provides three-dimensional, perspective views of the terrain and entity models. It is ideal for scenarios that require situational awareness, simulation debugging, or after action review (also known as AAR). It is particularly useful for placing entities with precise location and orientation and for working in terrains in which entities can be at multiple altitudes at a given X, Y coordinate, such as urban terrains.

The 3D view includes exaggerated reality (XR) features. Model scaling and additional model sets help to provide a common operational picture of the simulated world.

VR-Forces is compatible with the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) simulation standards. It supports multiple database formats and terrain servers. VR-Forces is interoperable with other MAK products, such as VR-Vantage Stealth, VR-Engage, and MAK Data Logger.

The VR-Forces GUI is built with the VR-Vantage Toolkit. The VR-Vantage Toolkit is based on OpenSceneGraph[1], so you can leverage value-added plug-ins built by the OSG community and MAK partners.

[1]"OpenSceneGraph is an Open Source, cross-platform graphics toolkit for the development of high-performance graphics applications such as flight simulators, games, virtual reality and scientific visualization. It is based on the concept of a SceneGraph, providing an object-oriented framework on top of OpenGL." (https://www.openscenegraph.com)

Table 1 lists VR-Forces visual features supported in the 3D and 2D views for entity-level scenarios.

Table 1. VR-Forces features by projection

Feature	3D	2D
Terrain agility and composability	X	X
Attach modes	X	X
Observer movement modes		
Observer frame	X	X
Level observer frame	X	
Laptop	X	X
2D level observer frame		X
Model sets		
3D models	X	X (top view)
3D colorized models	X	X (top view)
2D icons	X	X
Other Features		
Audio	X	X
Cockpit displays	X	
Communications lines	X	X
Detonation and muzzle flashes	X	
DI-Guy characters	X	
Display units configuration	X	X
Dynamic ocean	X	
Dynamic terrain	X	X
Emitter volumes	X	X
Entity labels	Text panels	Configurable symbol decorations
Environment effects	X	
Extruded tactical graphics	X	
Fire and detonation lines	X	X
Forward+ lighting	X	
High dynamic range lighting	X	
Illumination Flares	X	
Intervisibility	X	X
Paged terrain	X	X
Parachuting	X	X

Table 1. VR-Forces features by projection (continued)

Feature	3D	2D
Radar jamming	X	X
Range rings	X	X
Remote draw API	X	X (2D graphics only)
Sea spray effects	X	
Sensors	X	X (Stealth only)
Sensor contact lines	X	X
Shader-based effects maps	X	
Shadows	X	
SpeedTree trees	X	
Streaming buoys and beacons	X	X
Streaming terrain	X	X
Tactical Data Link support	X	X
Tactical graphics	X	X
Tactical smoke	X	
Task visualization	X	X
Terrain profiles	X	X
Tracers	X	
Track histories	X	X
Trailing effects	X	
View controls	X	X
Wireframe view	X	X
Zoom	X	X

1.1.1 Entity-Level and Aggregate-Level Simulation

VR-Forces supports two approaches to simulation – entity-level simulation and aggregate-level simulation. In entity-level simulations, VR-Forces models individual entities, such as ships, tanks, automobiles, airplanes, and people. Interaction, and for military simulations, warfare, takes place between individual entities. You can aggregate entities into hierarchical units, but for the most part they are still modeled as individuals.

Aggregate-level simulations do not model individual entity interactions and combat. They are concerned with the high-level movement and interaction of large numbers of personnel and equipment. Aggregate-level combat is modeled based on the relative strengths and weaknesses of hierarchical units in personnel, equipment, and location, not on the engagement of the individual entities that they represent. Aggregate-level simulations are used for modeling the operational tempo of large area/theater-level missions overseen by command staff level officers. This is useful for training staff officers and stimulating Command and Control systems (for example, C2, C4I, C4ISR, and Mission

Command systems). (Aggregate-level simulations include some entity models to facilitate certain types of munition attacks. However, they use the same modeling approach as units. They do not behave like the comparable entities in entity-level simulations.)

> Most of the information in VR-Forces documentation applies to both types of modeling. The procedures for creating scenarios, adding simulation objects and tactical graphics, assigning tasks and writing plans, and displaying scenario information are almost identical in both types of scenarios. Therefore, if you want to know how to create a scenario that uses aggregate-level modeling, for example, simply follow the procedures in "Chapter 7. Creating and Running Scenarios" on page 227. Or, if you want to learn about assigning tasks in entity-level scenarios, read "Chapter 29. Assigning Tasks" on page 557.
>
> If a section only applies to entity-level scenarios, it has an entity icon () in the margin. If a section only applies to aggregate-level scenarios, it has a unit icon () in the margin.

For conceptual information about the differences between entity-level modeling and aggregate-level modeling, see:

- "13.7. Entity-Level Modeling and Aggregate-Level Modeling" on page 368.
- "Chapter 25. Unit Behavior" on page 515.
- "Chapter 27. Aggregate-Level Modeling" on page 527.

1.1.2 Realistic Display of Vehicles and Terrain (3D View)

VR-Forces provides a realistic display using terrain and moving-model geometry in the OpenFlight format. It includes many 3D vehicle models, some of which display movement of articulated parts, such as turrets and rotors. These models can change to show a damaged or destroyed state. You can provide your own models and map them to object types. For details about terrain, see the chapter *Composing Terrains* in the *Adding Content to MAK ONE Applications Reference Manual*. For details about models, see the chapter *Mapping Models and Effects*, also in the *Adding Content to MAK ONE Applications Reference Manual*.

> Aggregate-level scenarios do not use 3D vehicle models.

1.1.3 Create Complex Scenarios

VR-Forces lets you create complex scenarios to meet your simulation requirements. You can:

- Create and delete simulation objects.
- Create, delete, and edit tactical graphics.
- Assign a sequence of tasks and set data requests to a simulation object as a plan.
- Assign tasks and set data requests independently of plans.
- Create global plans that run independently of any simulation object.

- Observe the behavior of simulation objects in response to their environment.
- Create, save, and load scenarios that consist of:
  - Terrain database information.
  - Simulation objects, such as individual platforms, humans, and units.
  - Tactical graphics.
  - A plan for each simulation object.
- Manage hostility relationships.
- Manage spot reports.
- Determine line-of-sight between simulation objects.
- Expand and collapse units.
- Create overlays and add tactical graphics and simulation objects to them.
- Schedule scenario events to inject non-simulated input, such as text, video, and graphics to scenario participants.
- Apply weather conditions to a particular area of the terrain.

1.1.4 Simulation Object Types Supported

Using VR-Forces, you can interactively add simulation objects to a simulation and you can aggregate them into higher echelon units. Simulation objects provided for entity-level scenarios include:

- Ground entities, such as:
  - Tanks
  - Utility vehicles
  - Artillery
  - IEDs
  - Civilian vehicles, such as automobiles, police cars, and fire engines
- Air entities, such as:
  - Fixed-wing aircraft, such as F-15 and F-35 jets
  - Rotary-wing aircraft, such as AH-64
  - Civilian aircraft, such as Boeing 737 and Airbus A330
- Surface and subsurface entities, such as:
  - Aircraft carriers
  - Frigates
  - Sailboats
  - Cargo ships
  - Submarines

- Lifeforms, such as:
  - Dismounted infantry
  - Civilians
  - Animals
- Units, such as platoons and companies.

Aggregate-level scenarios have simulation object types such as:

- ADA PLT
- Air Attack company
- Artillery battery
- Engineering company
- Ground attack aircraft
- Mechanized infantry company
- Mortar platoon
- Rifle company
- Tank company
- Destroyer
- Submarine

For a more comprehensive list of the simulation objects included with VR-Forces, see the *Entity Model Catalog*.

Simulation Object States

You can interactively modify aspects of a simulation object's state, including:

- Speed
- On-board stores, such as fuel and ammunition
- Location
- Heading
- Altitude

For more information about changing a simulation object's state, see "Section VII. Set Data Requests" on page 861.

You can modify simulation object parameters by editing the object parameter database and other configuration files.

Embarkation

Simulation objects can embark on and disembark from other simulation objects, for example, dismounted infantry can embark on a truck, HMMWVs can embark on an LCAC, and aircraft can land on and take off from aircraft carriers.

> To use embarkation in HLA federations, you must use RPR FOM 2.0, draft 17 or later. For information about specifying the correct FOM, see "4.1. Starting VR-Forces" on page 129 and "5.5.3 Specifying the RPR FOM Version" on page 191.

1.1.5 Mission Planning

You can create plans for simulation objects that contain tasks, set data requests, conditional statements and global commands. You can save plans and edit them.

Conditional statements specify that tasks will be executed only if a specific condition is true or false. Some of the conditions tested are:

- Is a simulation object in a specific area?
- Is a simulation object destroyed?
- Is a simulation object to the left or right of a phase line?

In addition to conditional statements that are checked only once (**If** statements), VR-Forces supports triggers, **While** statements, and **Wait Until** statements.

Global Plans

Global plans exist independently of any simulation object. They allow you to assign tasks and set data requests to simulation objects and execute other commands. In essence, they allow you to script the sorts of ad hoc actions that a VR-Forces user could take through the GUI. This provides a great amount of flexibility to automatically modify simulation objects and tactical graphics in response to events.

For more information, see "Chapter 43. Writing Plans" on page 947.

1.1.6 Simulation Object Tasks

You can assign tasks to simulation objects as part of a plan or independently. Some of the tasks supported in entity-level scenarios include:

- Move to an object.
- Patrol along a route.
- Patrol between two objects.
- Follow an entity.
- Fire for effect.
- Take off and land.
- Fly to a heading.

Some of the tasks supported in aggregate-level scenarios include:

- Move to an object.
- Patrol along a route.
- Patrol between two objects.

- Construct a minefield (and other combat engineering objects).
- Breach an obstacle.
- Bomb a location.

For a comprehensive list of tasks available to simulation objects, see "Section VI. Tasks" on page 553.

Independently assigned tasks override the tasks in a plan, allowing you to respond to the course of a simulation as events transpire without stopping to edit a simulation object's plan. For more information, see "Chapter 29. Assigning Tasks" on page 557.

1.1.7 Scripted Tasks and Set Data Requests

You can create new tasks and set data requests by writing scripts using the Lua scripting language. Input dialog boxes are created automatically. The scripted action can be added to the Task or Set menu. This feature requires some programming skills, but you do not need a developers license or a C++ development environment. For more information, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743.

Reactive tasks are a special kind of scripted task. They monitor the simulation and execute only if some special condition is met. You can manage assignment of reactive tasks in the GUI. For more information, see "29.5. Reactive Tasks" on page 572.

Background processes are another special type of scripted task. They run continuously and manage ongoing simulation object activities. There is no user interface for managing them. For more information, see "37.11. Background Processes" on page 812.

You can organize scripts into groups called Behavior Sets. By assigning Behavior Sets to forces you can control whether scripts are available to simulation objects in those forces. For more information, see "29.6. Using Behavior Sets to Manage Scripts" on page 577.

1.1.8 Tactical Graphics

Using VR-Forces, you can interactively create and name tactical graphics, such as:

- Points
- Phase lines
- Lines
- Areas
- Obstacles

Simulation objects are aware of some types of tactical graphics and can be assigned tasks such as following a route or moving between points. VR-Forces can also determine if specified simulation objects are in an area and this information can be used as part of mission planning.

Aggregate-level scenarios support combat engineering objects, which can be areal, point, or linear objects that could impede, damage, or defend a simulation object.

Other types of tactical graphics are for informational and tactical management purposes and do not affect the simulation. For more information, see "Section IX. Tactical Graphics" on page 991.

1.1.9 Terrain Agility and Composability

VR-Forces allows you to build your terrain at runtime using a variety of database and vector formats. It can load OpenFlight DTED, CDB, MetaFlight, and GDB (MAK terrain format) databases. It can load feature data from shapefiles, S-57 files, and GDB files. You can also superimpose geospatial images such as GeoTIFFs or other raster image files on the terrain.

You can load local terrain data and you can stream data from networked servers using the osgEarth earth file format.

> VR-Forces only supports geocentric terrains in osgEarth earth files.

You can save these composed terrains in the MTF file format. For details, see the chapter *Composing Terrains* in the *Adding Content to MAK ONE Applications Reference Manual*. For a list of supported file formats, see "63.2. Supported File Formats" on page 1242.

VR-Forces supports these coordinate systems:

- UTM
- Geocentric
- MGRS
- Latitude, longitude
- Database

You can overlay raster images, such as CADRG, BMP, and GeoTIFF files on terrain databases.

1.1.10 Dynamic Terrain

Terrain state can be changed using switch nodes in the terrain models. Terrain changes can be caused by munition damage, user initiated changes, and using entity tasks.

Cultural features can be embedded in the terrain and navigation data can be dynamically regenerated so that entities can navigate around and through the objects.

In scenarios, users can place terrain control objects to alter the terrain and the way entities move through it.

1.1.11 Crowd Behaviors and Pattern of Life

You can quickly create civilian crowds that have inherent behaviors of wandering and fleeing munition detonations. You can assign tasks to crowds, such as gathering around a person or point. For details, see "Chapter 17. Crowds and Herds" on page 431.

You can use the pattern of life feature to generate civilian and vehicular traffic using various spawn patterns to simulate typical background traffic for scenarios. For details, see "Chapter 16. Generating Traffic (Pattern of Life)" on page 415.

1.1.12 Flexible, Intuitive Graphical User Interface

VR-Forces's graphical user interface (GUI) allows you to manipulate VR-Forces from menus, dockable control panels, and the keyboard.

The VR-Forces display includes these features:

- Manipulate the simulation map by zooming and panning.
- Select and track simulation objects.
- Turn display features on and off.
- Display a hierarchical, echelon view of simulation objects and an embarkation view.
- Display entity labels that include several types of information.
- Display line-of-sight (intervisibility) for simulation objects.

You can hide and display the control panels, keep them docked to the screen, or undock them to make more space available for the display window. The GUI controls can be hidden for full-screen demonstrations.

The GUI follows standard conventions for modern windowing systems. For additional details, see "Chapter 84. GUI Controls, Layouts, and Behaviors" on page 1621.

1.1.13 Overlays

Overlays mimic clear plastic sheets that a planner might use to draw graphics on top of a terrain map. All objects are automatically assigned to overlays. You can add overlays and move objects between overlays. You can hide the display of objects on an overlay. For details, see "Chapter 46. Overlays" on page 1003.

1.1.14 Behaviors

Simulation objects understand how to interact with their environment. For example, they can:

- Respond appropriately to other simulation objects and tactical graphics, for example, avoiding simulation objects, waiting for orders, and so on.
- Respond to detected targets, for example, deciding which of multiple targets to acquire, which weapon to fire, and what ammunition to use.
- Follow a route.

1.1.15 Observer Attach Modes

The observer represents your eyepoint in the simulated terrain. VR-Forces offers attach modes that let you:

- Manually control the observer's position and orientation. (The observer represents the user's view, or eyepoint, in the simulated world.)
- Follow a simulation object, maintaining a consistent positional offset, with a matching heading.

- Mimic a vehicle's position and orientation. You can place the observer inside the vehicle for a driver's-eye view, or fly outside of it, moving in tandem with the vehicle.
- Tether the observer to a simulation object, maintaining a consistent positional offset without changing the observer's heading.
- Automatically track a vehicle's movement from a fixed viewpoint, as if watching from a control tower.

For more information, see "Chapter 60. Moving the Observer" on page 1175.

1.1.16 Special Effects and Simulation Object Information Visualization

VR-Forces includes features that help you understand the interaction between simulation objects that add a realistic tone to interactions. These features include:

- Fire and detonation lines.
- Track histories.
- Range rings.
- Radio communications lines.
- Trailing effects, such as footprints, tire tracks, and wakes.

1.1.17 Dynamic Ocean

VR-Forces provides dynamic ocean visualization in the Stealth and Out-the-Window observer modes. The ocean shows waves, swells, and spray effects. Surface entities have realistic wakes and buoyancy behavior. Changes to wind speed and direction affect wave action. For more information, see "11.8. Marine Conditions" on page 341.

1.1.18 Lighting Effects

VR-Forces supports a variety of lighting effects, including:

- Dynamic lighting
- High dynamic range (HDR)
- Horizon-based ambient occlusion
- Ocean planar reflection
- Lens flare
- Crepuscular (sun) rays

Shader-based Effect Maps

VR-Forces supports several types of shader-based effect texture maps. Shaders are computer programs that simulate the effect of light on the terrain and objects in the simulation. VR-Forces makes extensive use of them for visualization effects.

VR-Forces supports these types of shader-based effect maps:

- Normal, or bump, maps. Normal maps give terrains the appearance of relief, such as a rocky landscape.
- Specular maps. Specular maps affect the highlight color of objects.
- Ambient occlusion maps. Ambient occlusion maps model areas that do not receive direct light, such as cracks and crevices and shaded areas of terrain and models. These areas are lit only by ambient light.
- Reflection maps. Reflection maps affect the reflectivity of surfaces, such as windows. Reflection maps reflect objects in the sky, not the terrain.
- Emissive maps. Emissive maps control the emissivity of whatever they are applied to based on the current ambient light values.

Lighting effects are described in "Chapter 52. Lighting Effects" on page 1087.

1.1.19 Accurate Vehicle Positioning

VR-Forces can smooth the trajectories of moving vehicles to compensate for discontinuous positional data. The ground clamping feature can ensure that all ground entities are placed correctly on the terrain surface. For details, see "6.10. Trajectory Smoothing" on page 213 and "63.7. Using Ground Clamping" on page 1248.

1.1.20 Batch Mode Operation

Batch mode allows you to run a scenario multiple times or run several different scenarios, without direct action through the graphical user interface. This is useful for running unattended testing or to test the results of a scenario under differing conditions. You can record the results of batch mode simulations using the MAK Data Logger, for later review. For more information, see "7.10. VR-Forces and Batch Mode" on page 269.

1.1.21 Remote Control

You can control the VR-Forces observer remotely through view control messages, which can be generated by MAK applications such as MAK Data Logger, and by programs that use the VR-Vantage Control Toolkit. For details, see "60.14. Controlling the Observer from Other Applications" on page 1207.

You can also send view control messages from plans. For details, see "43.7.7 Sending View Control Messages" on page 975.

1.2. The VR-Forces Toolkit

The VR-Forces toolkit lets you develop CGF applications. You can enhance the capabilities of the VR-Forces simulation engine and GUI, or integrate the simulation engine into your own user interface. The toolkit has several APIs. The Simulation API gives you control over:

- Behaviors
- Components

- Object types
- Parameters
- Messages
- Resources
- Tactical graphics
- Plans
- Tasks
- SQL database logging

The Remote Control API allows you to control a VR-Forces simulation engine from a remote application, such as a graphical user interface (GUI). The Terrain API lets you manipulate terrain databases.

The VR-Forces GUI is built with the VR-Vantage Toolkit. The toolkit is included with VR-Forces, allowing you to modify or extend the GUI. For details about the VR-Forces Toolkit, please see the *VR-Forces Developers Guide*.

1.2.1 Plug-in Architecture

You can modify the VR-Forces GUI and sim engine by rebuilding the applications or by writing plug-ins. In fact, much of the basic functionality of the GUI is implemented using plug-ins rather than as part of the main executable. Plug-ins are dynamically linked libraries that VR-Forces can load at startup. The rationale for choosing which method to use is described in API documentation. You can manage the use of plug-ins in the GUI. For details, see "4.9. Managing Plug-ins" on page 156.

1.2.2 SQL Database Logging

Developers can configure VR-Forces to save data to a database, recording what happens in a simulation (who saw whom, who shot whom and how long after detection, which resources they used, what was destroyed, and so on). After users run a VR-Forces scenario, developers can use that data for analysis. They configure this using the *./appData/settings/databaseConfig.mtl* file, which includes comments explaining how to use it.

SQLite is supported as the built-in database implementation. MySQL is supported as a plug-in example. SQLite records a single file per simulation engine. While much of the internal state data is recorded, you can add additional data and events using C++ and Lua. If you have multiple sim engines, developers can use open source tools to merge multiple SQLite databases together, if desired. See the SDK toolkit examples addDatabaseEvent, databaseClients, and databaseConnection.

To learn more about saving data to a database, see "Database Logger" in the *VR-Forces Developers Guide* (found under "2. Simulation Services" in the "Miscellaneous" section).

1.3. DI-Guy

VR-Forces uses DI-Guy software and content for human character animation.

VR-Forces comes with DI-Guy functionality built-in and with a large set of DI-Guy characters, appearances, and animations. A VR-Forces customer does not need a DI-Guy license of any kind to use DI-Guy animated characters in a VR-Forces-based application (including custom applications built using the VR-Forces Toolkit).

The DI-Guy Character Viewer is bundled with VR-Forces. It lets you view the entire library of DI-Guy characters, hand items, heads, and animations. For details about using it, please see the *Adding Content to MAK ONE Applications Reference Manual*.

However, you need a DI-Guy Developer's license if you want to:

- Develop new character types, or alter the geometry, appearances, textures, or motions of DI-Guy characters.
- Directly access the DI-Guy API to modify or extend the way that VR-Forces uses the DI-Guy API or access DI-Guy functionality that MAK has not already integrated into VR-Forces.

DI-Guy libraries, models, textures, motion files, and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of DI-Guy software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

For more information, see "32.13. DI-Guy Animation" on page 654, "40.25. DI-Guy Characteristics (Appearance, Head, Body Weight)" on page 876, "74.2. DI-Guy Configuration Files" on page 1464, and "70.1.10 Editing a Lifeform's Visual Model and Animation" on page 1383.

1.4. Helpful Utilities

VR-Forces includes these utilities for configuring and troubleshooting VR-Forces:

- The Simulation Object Editor is a graphical user interface that you can use to add and remove component systems from simulation objects and specify common simulation object parameters. You can create new simulation objects in the editor. You can also create new simulation model sets. The Simulation Object Editor is also used to manage tactical graphics. For details, see "Chapter 68. The Simulation Object Editor and SMSs" on page 1303.
- The vrfMsgDump utility displays the contents of VR-Forces messages discovered on the network. For details about vrfMsgDump, see the *VR-Forces Developers Guide*.
- The Boundary Generation Tool calculates the boundaries of a high resolution terrain inset for inclusion in an earth file. For details, see "Using the Boundary Generation Tool" in the *Adding Content to MAK ONE Applications Reference Manual*.
- The medfTool enables encrypts 3D model files. For details, see "Compressing Model Files" in the *Adding Content to MAK ONE Applications Reference Manual*.

- The mftTool processes MetaFlight terrains so that they will work in VR-Forces. For details, see "The mftTool" in the *Adding Content to MAK ONE Applications Reference Manual*.
- The osgEarth cache feature caches terrain data from earth files offline. For details, see "Caching earth File Terrain Data Offline" in the *Adding Content to MAK ONE Applications Reference Manual*.
- The vectorNetworkToShapeFiles utility extracts feature data from GDB, DFD, and VMAP into shapefiles, which is the preferred format for adding features to terrain. For details, see "Extracting Feature Data from GDB to Shapefiles" in the *Adding Content to MAK ONE Applications Reference Manual*.
- The translationFileCreate utility extracts the user-visible text in a script dialog box to make it available for translation. For details, see "2.6. Localizing the Graphical User Interface" on page 108.
- The VR-Forces User Settings Cleanup Tool resets your settings to factory defaults.

1.5. Third-Party Software and Content

VR-Forces includes software and content licensed from third parties, including:

- SilverLining™: real-time sky and 3D cloud rendering from Sundog Software.
- Triton Ocean SDK™: 3D ocean and ship wakes from Sundog Software.
- GL Studio®: interactive cockpit instrumentation and HMI from DiSTI®.
- SpeedTree®: animated, 3D foliage from Interactive Data Visualization (IDV).
- 3D models, terrain, and graphical content from Simthetiq, RealDB, TurboSquid, and TerraSim.
- OpenSceneGraph: an open source 3D graphics toolkit hosted at https://www.openscenegraph.com.
- osgEarth: an open source streaming terrain plug-in by Pelican Mapping, at http://osgEarth.org.

> ⚠ The run-time and developers' rights for VR-Forces customers vary from vendor to vendor. Please read this section carefully to understand which rights are included with VR-Forces licenses, and which rights require additional third-party licenses.

1.5.1 SilverLining

VR-Forces uses SilverLining software and content to compute lighting and to render the sky, clouds, sun, and moon. SilverLining is developed by Sundog Software (https://www.sundog-soft.com).

Most VR-Forces customers do not need to buy a SilverLining license of any kind. In general, a VR-Forces customer has the right to use the SilverLining technology that is built into VR-Forces in any VR-Forces-based application (including custom applications built using the Toolkit). However, there are two exceptions:

- If you want to extend or change the way that MAK has integrated SilverLining into VR-Forces by writing directly to the SilverLining API, then you need a SilverLining Developer's License.
- If you are building a commercial video game or other "mass-market" (consumer-level) commercial product on top of the VR-Forces Toolkit, and the resulting product includes the SilverLining functionality, then you need a SilverLining Developer's License.

By default, SilverLining (and therefore VR-Forces) uses a hard-coded star catalog. You can override this by setting the **star-catalog-file** parameter for the star catalog in the config file, *./data/Environment/Sky/SilverLining.config*. The comments include a reference to the example file *starcat.csv*, also located in the *Sky* directory; it shows the format SilverLining expects for the catalog.

SilverLining libraries, textures and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of SilverLining software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.2 Triton Ocean SDK

VR-Forces uses the Triton Ocean SDK to render 3D oceans and ship wakes. Triton Ocean is developed by Sundog Software (https://www.sundog-soft.com).

Most VR-Forces customers do not need to buy a Triton Ocean license of any kind. In general, a VR-Forces customer has the right to use the Triton Ocean technology that is built into VR-Forces in any VR-Forces-based application (including custom applications built using the Toolkit). However, there are two exceptions:

- If you want to extend or change the way that MAK has integrated Triton Ocean into VR-Forces by writing directly to the Triton Ocean API, then you need a SilverLining Developer's License.
- If you are building a commercial video game or other "mass-market" (consumer-level) commercial product on top of the VR-Forces Toolkit, and the resulting product includes the Triton Ocean functionality, then you need a Triton Ocean Developer's License.

Triton Ocean libraries, textures and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of Triton Ocean software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.3 GL Studio

VR-Forces uses GL Studio software and content to render interactive cockpit instrumentation displays. GL Studio is developed by DiSTI (https://www.disti.com).

VR-Forces is delivered with several generic cockpit displays. You can use the built-in cockpit displays in any VR-Forces-based application (including custom applications built using the Toolkit), without a GL Studio license of any kind.

If you want to build custom cockpits using the GL Studio editor or GL Studio API, you need a GL Studio Developer's license. If you want to execute custom cockpits in VR-Forces-based applications (regardless of whether the custom cockpits are built by you or a third party), you need GL Studio Run-Time licenses.

GL Studio libraries and graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of GL Studio software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.4 SpeedTree

VR-Forces uses SpeedTree software and content for animated real-time 3D foliage and vegetation. SpeedTree is developed by Interactive Data Visualization (IDV) (https://speedtree.com).

A VR-Forces customer does not need a SpeedTree license of any kind to use the SpeedTree functionality that is built into the VR-Forces applications that MAK delivers. You may build and run plug-ins to our out-of-the-box applications without a SpeedTree license, as long as your plug-ins do not need to directly access the SpeedTree API.

However, you need a SpeedTree Developer's license if you want to:

- Write a plug-in that uses the SpeedTree API directly to extend or modify the way that SpeedTree is integrated into VR-Forces.
- Use the VR-Forces Toolkit to develop new applications, and you want your applications to include SpeedTree functionality.

SpeedTree libraries, models, textures, and other graphics resources are distributed in the VR-Forces package solely so that VR-Forces can use them. Use of SpeedTree software or content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.5 3D Models, Terrain, and Graphical Content

VR-Forces includes a rich set of 3D models for vehicles, weapons, cultural features and urban clutter (signs, barriers, lampposts, and so on). It also includes several sample terrain databases to help you get started with VR-Forces, and to help demonstrate VR-Forces. Much of this content is derived from 3D data that we have licensed from third parties:

- Many of the high-quality vehicle models come from Simthetiq (https://simthetiq.com).
- Some of the middle-eastern building models and urban clutter objects that are used in our sample terrain databases were licensed from Turbosquid (https://turbosquid.com).
- Some models come from Real DB.

All of this content is distributed in the VR-Forces package solely so that VR-Forces can use it. Use of any of the VR-Forces models, textures, terrains, or other content outside of VR-Forces-based applications is not permitted by the VR-Forces license.

1.5.6 OpenSceneGraph

VR-Forces uses OpenSceneGraph for its underlying scene graph representation, rendering, file loading, and many other capabilities. OpenSceneGraph is an open source, cross-platform graphics toolkit for the development of high-performance graphics applications. The OpenSceneGraph source repository is maintained by Robert Osfield at https://openscenegraph.com. It is distributed under the OpenSceneGraph Public License (OSGPL), which is based on the GNU Lesser General Public License (LGPL).

MAK will provide modified source code for OSG upon request, as per the OSGPL license. Please contact MAK support at https://www.mak.com/mak-one/support/portal to obtain the download links for our modified source.

1.5.7 osgEarth

VR-Forces uses osgEarth to import streaming terrain elevation and imagery data. The data can be streamed from external servers and sources or from a directory on the computer running VR-Forces. osgEarth is an open source plug-in to OpenSceneGraph, maintained by Pelican Mapping at http://osgEarth.org. osgEarth is distributed under the GNU Lesser General Public License (LGPL).

MAK will provide modified source code for osgEarth upon request, as per the OSGPL license. Please contact MAK support at https://www.mak.com/mak-one/support/portal to obtain the download links for our modified source.

1.6. Distributed Simulation Standards Supported

VR-Forces supports the and HLA Evolved (IEEE 1516-2010) specification. It also supports the Distributed Interactive Simulation (DIS) protocol version 7.

VR-Forces has built-in support for the HLA RPR-FOM and can support other FOMs through the FOM Mapping feature. For information about FOM mapping, see "5.5.4 Specifying a FOM Mapper" on page 191. Please see the *VR-Forces Release Notes* for a list of the versions of DIS supported.

VR-Forces supports time management for HLA exercises.



2. Installing VR-Forces

This chapter explains how to install VR-Forces on your computer. It also explains how to install other software required for use with VR-Forces.

2.1. Installing VR-Forces 96
- 2.1.1 Installation and Data Packages 96
- 2.1.2 Installing VR-Forces on Windows 96
- 2.1.3 Installing VR-Forces on Linux Systems 97
- 2.1.4 Installing VR-Forces PDB Files for Debugging Purposes 98
- 2.1.5 Installing the Toolkit 99
- 2.1.6 The makData Package 100
- 2.1.7 Uninstalling from Windows 102
- 2.1.8 Uninstalling from Linux 102

2.2. The VR-Forces Directory Structure 102
2.3. Installing and Setting Up the MAK License Manager 104
- 2.3.1 Specifying the License Server 105

2.4. Installing an RTI 106
- 2.4.1 Installing the MAK RTI 107

2.5. Using MAK ONE Products on Virtual Machines 108
2.6. Localizing the Graphical User Interface 108
- 2.6.1 Translating Other Interface Files 110
- 2.6.2 Translating VR-Forces Scripts and Console Messages 111
- 2.6.3 Applying the Language Files 112

2.1. Installing VR-Forces

2.1.1 Installation and Data Packages

MAK ONE installations are provided in one or more files (packages) that you can download from the MAK website.

There are four types of installation products for MAK ONE:

- Product installers – Each product has a single main installer for each operating system.
- Optional installers – Some products have installers for developer options.
- Required data packages – MAK ONE applications (VR-Vantage, VR-Forces, VR-Engage, SensorFX, and RFView for MAK ONE) and the DI-Guy SDK have a collection of data packages that are required for those products' basic functions.
- Recommended data packages – Additional data packages (if available) are recommended to access the full capabilities of MAK ONE applications.

MAK ONE applications share the same data installation, therefore the same set of data packages is all you need for any or all applications (VR-Vantage, VR-Forces, VR-Engage, SensorFX, RFView for MAK ONE, and the DI-Guy SDK).

Dependencies

Licensing tools and dependent products have their own installers. For details, see "2.3. Installing and Setting Up the MAK License Manager" on page 104.

For interoperability using HLA, see "2.4. Installing an RTI" on page 106.

Before you install VR-Forces, please read the release notes to see if there are any special instructions for installation.

Build Information

If you need to know the details about the MAK ONE application you are using, you have two options:

- While running the application, choose **Help > About** and then read the information in the About dialog.
- In the installation directory, look at a build information text file, for example, VR-Forces_Build_Information.txt.

2.1.2 Installing VR-Forces on Windows

Windows versions of VR-Forces are provided as downloaded files.

➤ To install VR-Forces, run the installer. Follow the instructions in the installation wizard. (You install the data used by VR-Forces separately, as described in "2.1.6 The makData Package" on page 100.)

Note the following:

- The installers are named to indicate the compiler used to build that version of VR-Forces.
- When you install large applications on Windows, there may be a delay of up to several minutes from the time you try to run the setup program to the time that an installation dialog box displays. This is due to how Windows scans setup programs before it executes them. If you experience this problem, turning off User Access Control can reduce or eliminate this delay.

Default Windows Installation Directory

The default root directory for installing MAK ONE products is *C:/MAK*.

You can use an environment variable, MAK_INSTALL_DIR, to specify a different location where all MAK ONE products and data should be installed by default. For example, if you define it as MAK_INSTALL_DIR = D:\MyProject, then all MAK ONE products automatically install to that location. The data packages and products would then install in that directory by default: *D:/MyProject/SharedData*, *D:/MyProject/vrlink5.10*, and so on.

Installing the Product Documentation

Due to the limitations of Windows installers, the user and developer documentation for VR-Forces are in a separate installer. To access the documentation locally, you must run the product's **.docu.exe* installer and install the files in the product install directory.

➤ **To install the documentation:**

1. Run the product's documentation installer, *vrforces#.#.docu.exe*.
2. The wizard finds the location where you installed the product. Accept that location to ensure that links to the documentation work correctly.
3. Optionally, change the Start menu location for the shortcuts to the documentation. If you want, you can choose a location separate from product installation directory. For example, you might want to put MAK ONE documentation shortcuts in a central location. This change does not affect the documentation shortcuts folder that is already located in the product installation directory.
4. Complete the installation wizard. The product documentation will be available on your machine.

2.1.3 Installing VR-Forces on Linux Systems

Linux versions of VR-Forces are provided as compressed TAR files.

➤ **To install a product from a TAR file:**

1. Create the directory in which you want to install the product.
2. Copy the tar file to the install directory.
3. Uncompress and untar the tar file:

   ```
   tar -xzf applicationxx-os.tar.gz
   ```

   where *xx-os* is the release, operating system, and other product release coding.

2.1.4 Installing VR-Forces PDB Files for Debugging Purposes

In cases where a MAK ONE application crashes, it can be difficult to determine the cause. The application generates a *.dmp* file that contains valuable information about the application's state at the time of the crash. MAK technical support can analyze this file to help diagnose the issue.

In addition, with the application's PDB (program database) files, the application generates a call stack log file (*.callstack.log*) of the application at the time of the crash. The call stack log provides information in plain text about the sequence of functions that were called leading up to the crash, which is also helpful in identifying the specific cause.

By providing the *.dmp* or *.callstack.log* files to technical support, you help support personnel to resolve the issues. They use specialized debugging tools to analyze the files and determine the cause of the crash, and that information aids in the development of a solution to prevent future crashes.

Due to security reasons or organization policies, some customers cannot send the *.dmp* file to technical support. Because the call stack log file is a plain text file, it is easier to inspect and analyze than the *.dmp* file. Users can easily attach it to an email and pass it out of secure areas with visual inspection, making it a convenient option for customers with restrictions on sharing binary files.

It is important to note that the call stack log file alone might not be sufficient for support personnel to fully diagnose the issue. They may require more information to gain a complete understanding of the issue. A detailed procedure for reproducing the crash often helps them solve the problem you encountered.



- Support for the generation of the call stack log file was introduced in MAK ONE 2024. It is supported only in MAK ONE products part of that release and later versions.
- The PDB files are part of the Linux installation. You do not need to install them separately.

Installing PDB Files on Windows

Before installing the PDB files on Windows, you must install VR-Forces. Next, make sure that the PDB files you are installing are from the same build as the application. When you download the installer from the MAK ONE Product Downloads page, the appropriate PDB installer is also available for download.

➤ **To install the PDB files:**

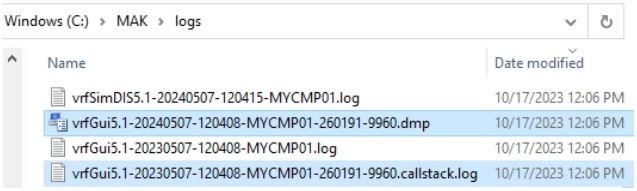
1. After installing VR-Forces, run the PDB installer for that product.
2. The installer looks for the matching product directory by searching the registry and then it automatically sets the install directory to the appropriate path. If that is not correct for your product, see the note below.
3. Complete the installation wizard. The necessary PDB files are installed in the product's *./bin64* directory.

- The default installation directory is *C:\MAK*. See "Default Windows Installation Directory" on page 97 to learn about setting the default to other locations.
- Be sure that the folder in the Select Destination Location page of the installation wizard is correct. It should be the top-level folder for the product installation.
- If the PDB installer displays a warning that the SVN revisions do not match, make sure that you have the correct PDB installer for the version of VR-Forces that you installed.

Reporting an Issue Using the *.callstack.log* and *.dmp* Files

Send the *.callstack.log* and *.dmp* files (or only the *.callstack.log* if you are unable to provide the *.dmp* file) to MAK support through the Support Portal (https://www.mak.com/mak-one/support/portal) or support@mak.com. The files will have similar names and are located in the central directory for MAK ONE log files (Figure 1). Please also provide information on anything you experienced when the issue occurred.

Figure 1. Example of *.callstack.log* and *.dmp* files for VR-Forces



Specifying the Location for *.log* and *.dmp* Files

The default directory for MAK ONE log files is *C:/MAK/logs*. You can customize the location by specifying a different path in the MAK_LOG_DIR environment variable.

2.1.5 Installing the Toolkit

The SDK toolkit is an API that allows developers to create and add functionality to VR-Forces, and it also includes examples about how to use the SDK.

The VR-Forces installer includes an option to install the toolkit. This requires that you purchase a separate license, which comes with its own installation code.

If you skipped the toolkit installation when you installed VR-Forces but you now have a toolkit installation code, you must rerun the application installer and enter the code when prompted. Be sure to install it in the correct directory. In the Select Components page of the installation wizard, simply clear the runtime application check box and select the toolkit check box.

Required Software for Toolkit Users

If you plan to build VR-Forces applications using the VR-Forces Toolkit, you must download and install VR-Link 5.10. If you are building for HLA and want to link with the MAK RTI, use MAK RTI 3.x or later, although we recommend use of the latest release. If you are building for HLA Evolved, you must use MAK RTI 4.0.4 or later.

To build VR-Forces applications, you must link with VR-Link 5.10. If you plan to build the VR-Forces GUI, you must also download and install Qt 5.15.2 (Windows VC++ 15 and Linux Red Hat 8). You can download it from the Bonus Downloads page on the website (https://www.mak.com/mak-one/support/help/bonus-material).

Qt is also available as a free download under the LGPL version 2.1 license at https://www.qt.io. This version should be satisfactory for most VR-Forces customers. If you need a Qt commercial license, you must purchase the license from Qt.

2.1.6 The makData Package

The MAK Data Packages are used by multiple MAK ONE applications. The applications need to be able to find the data, wherever you install it. VR-Forces looks for data in a default location or uses a collection of environment variables to find the data in the location where you installed it. You have several options for installing the data and telling VR-Forces where to look for it.

MAK Data Installation Directory

The shared data files are stored in a directory file hierarchy. This structure allows for multiple versions of data to be installed for use with different versions of VR-Forces.

The directory looks like *shared-data-root/data-compatibility-version/stem/data-folders*, where:

- *shared-data-root* is the location of the shared data directory. The default is *C:/MAK/SharedData* on Windows and *~/MAK/SharedData* on Linux.
- *data-compatibility-version* is a number, such as 18 or 19.
- *stem* is the directory holding the data files, for example, *latest* for the most recent installation of that compatibility version or the date of a previous installation, such as *2024-03-14*.
- *data-folders* hold the various files that make up the MAK data files (models, audio files, fonts, and so on).

Advanced Windows users may want to install MAK Data Packages in a very specific way, for example, installing the data on a shared drive and configuring any machines with MAK ONE applications to look there for the data. Refer to the *MAK ONE Installation Guide* for instructions.

Installing the makData Package

> ⚠ Currently, the MAK ONE Software Manager does not work on Red Hat 9. On that OS, you must extract the data manually to the correct location.

➤ **To install the data:**

1. Install the MAK ONE Software Manager for your operating system.
2. Run the MakOneSoftwareManager executable in the *.\bin64* folder.

   On Windows, you can also open the Start menu and then choose **MAK Technologies > MAK ONE Software Manager**.

3. Click Select and Install Data Packages in the welcome page.
4. Specify the directory where you saved the data files. The default is your Downloads folder, but you can click the folder icon to navigate to another folder.
5. Select the data files that you want to install.

   MAK Technologies recommends that you install all of the data files if you have the space and are not absolutely sure which files you require.

   You must install the Core data for VR-Forces to function. To have access to other terrains and to see anything other than notional models, install the other data packages provided by MAK. If you do not install any other data, it is assumed that you are providing your own models and terrains.

6. Optionally, choose the location where you want to install the data.

   MAK recommends that you install the files in the default location. While MAK ONE products allow for diverse installations, the typical use is the standard installation. If you require a more complex installation, refer to the *MAK ONE Installation Guide*.
7. If you have a previous installation of data for the same compatibility version, you are prompted to choose whether to back up the previously installed data, delete it, or overwrite it. See "Options When Installing a Data Package" below.
8. Click Install. The MAK ONE Software Manager shows the progress of the data installation.
9. When the installation is complete, click Exit.

Options When Installing a Data Package

When installing a data package using the MAK ONE Software Manager, if you have no preexisting installations of that compatibility version, it simply installs the data files in the *./SharedData/##/latest* folder (where *##* is the compatibility version).

If there is already a previous release of the same compatibility version installed, however, you are prompted to choose whether to back up, delete, or overwrite the data from the previous release.

- Backup. If you install a newer release and choose to create a backup of the previous release, the files in the *latest* directory are moved to a new directory named for the data version's date in the same compatibility folder and the new files are then installed in the latest folder. For example, the old files move to *./SharedData/18/2024-03-14* and the new files are installed in *./SharedData/18/latest*.

  Creating a backup is the recommended option in most circumstances. It allows all MAK ONE applications to use the newest data, while also giving you the option to revert to a previous release using command-line options or setting a Windows environment variable. See the *MAK ONE Installation Guide* for details.
- Delete. The installer deletes the old data from the *latest* directory and then installs the new data.

- Overwrite. The installer overwrites the previous data files in the *latest* directory with the newer ones you chose to install. It also adds files that are new and ignores files that were previously installed if the package you are installing does not include that file.

2.1.7 Uninstalling from Windows

MAK ONE product installations include an uninstaller application (*unins000.exe*), which you can find in the installation directory. Run that program to uninstall the product.

When you run other installers for the product, such as the PDBs and documentation, there are additional uninstallers. If you run the uninstallers from the highest number down (for example, *unins002.exe* and then *unins001.exe*), running *unins000.exe* uninstalls everything else and removes the install folder.

When you uninstall VR-Forces, files such as scenarios or SMSs that reside in the product directory are also deleted. Before uninstalling, copy any of your custom files from the product directory to another folder.

The uninstaller does not remove the data package. That is in a separate directory (for example, *./SharedData/19/latest*).

> If you run only the product uninstaller, you can then simply delete the rest of the product's install folder. Note that if you do not run all of the uninstallers, running only the product uninstaller may not remove all of the shortcuts from the Windows Start menu.

2.1.8 Uninstalling from Linux

To remove VR-Forces installations from Linux machines, simply delete the installation directory. This catches all of the files associated with the product.

When you uninstall VR-Forces, files such as scenarios or SMSs that reside in the product directory are also deleted. Before uninstalling, copy any of your custom files from the product directory to another folder.

Deleting the installation directory does not remove the data package. That is in a separate directory (for example, *./SharedData/19/latest*).

2.2. The VR-Forces Directory Structure

The VR-Forces installation process creates a directory structure like the one shown in Figure 2. On PCs, the default installation directory is *C:\MAK\vrforces#.#.*

Figure 2. VR-Forces directory structure

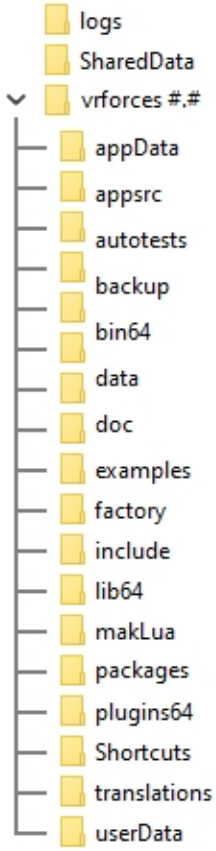


Table 2 describes the contents of the directories installed by VR-Forces.

Table 2. Contents of VR-Forces directories

Directory	Directory contents
./appData	Schema, model definitions, settings, configuration files, and other data.
./appsrc	Source files and project files for API developers.
./backup	Backup configuration files.
./bin64	Product and utility executables, dynamic link libraries (DLLs). Also where PDBs are installed.
./data	The *./data* directory has several subdirectories that contain: • Bitmaps and icons • Simulation model sets • Terrain database source files • Visual model data • Other data required by the application

Table 2. Contents of VR-Forces directories (continued)

Directory	Directory contents
./doc	VR-Forces documentation. This is the user documentation, both PDF and HTML5, as well as class documentation for users who have the VR-Forces toolkit. Please note that the documentation set (including the *VR-Forces Developers Guide*) are in an installer separate from the product installer.
./examples	Sample code for toolkit users.
./factory	Backups of the application data and settings provided by MAK.
./include	Header files for use by API developers.
./lib64	Library files.
./packages	Header files and libraries for third-party and supplementary APIs that developers may need to build against.
./plugins64	Plug-in DLLs.
./sfx	Sound effect files
./Shortcuts	Contains the shortcuts to the documentation and tools, which are used for the Windows Start menu.
./translations	Files and an executable for localizing the GUI.
./userData	Scenarios, terrain files, and other files that are created as you use VR-Forces.

2.3. Installing and Setting Up the MAK License Manager

Before you can use a MAK ONE product, you must obtain a valid license file, install the MAK License Manager, and configure the license server and client machines. The License Manager uses a client-server architecture, so you do not need to install the License Manager on every computer on which you install MAK ONE products. You only need to install it on the computer that you will use as the license server.

> If you have already installed the License Manager for another MAK ONE product, you do not have to install it again. You only need licenses for your newly installed products.

The License Manager installer is included with MAK installation media. It is separate from the product installers. You can download the installers from the MAK ONE Product Downloads page at https://www.mak.com/mak-one-downloads.

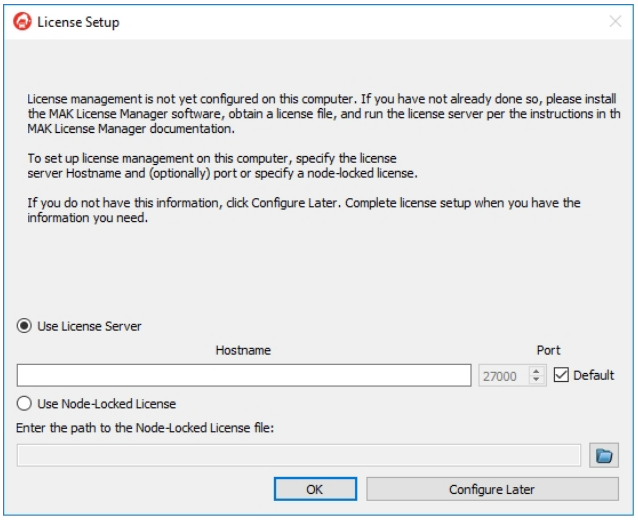
Complete installation and configuration instructions are included with the License Manager installer. The *MAK ONE Installation Guide* is available at https://mak.com/mak-one-installation-guide.

Some customers use dongle licenses instead of running a license server. Instructions for using dongles are also in the *MAK ONE Installation Guide*.

2.3.1 Specifying the License Server

The first time you run a MAK ONE product on a particular computer, the License Setup dialog box opens (Figure 3). It prompts you to enter the hostname of the license server and, optionally, a port number.

Figure 3. License Setup dialog box



If you do not know the hostname of the license server, click Configure Later. When you have the hostname, you can start the application again and complete the dialog box. You will not be able to run any MAK ONE products until you set up license management.

If you know the hostname, enter it in the Hostname box and then click OK. The application will start.

Under limited circumstances, MAK issues node-locked licenses. If you have a node-locked license, select the Use Node Locked License option and enter the path to the license file.

- If you are running MAK ONE products on the license server machine, it is also a client, so you must specify the license server on that machine, too.
- If you change the license server, the saved configuration will no longer be valid and the License Setup dialog box will open the next time you start a MAK ONE product.
- If you have changed the license server, you may want to clear the license cache. You can clear the saved license configuration by deleting the cache file. On Windows, it is *C:\Users\user_name\AppData\Roaming\MAK\licenses<n>.txt*. (The *AppData* directory is hidden by default.) On Linux, it is *.mak/license<n>.txt*. (There may be more than one cache file, for example, *licenses1.txt* and *licenses2.txt*.) You can also clear the cache in the RTI Assistant.

Setting the MAKLMGRD_LICENSE_FILE Environment Variable

As an alternative to using the license setup procedure described in the previous section, you can configure the license server or node-locked license in an environment variable. The MAKLMGRD_LICENSE_FILE environment variable identifies the server machine. If you

set this environment variable, it overrides the settings stored by the license setup procedure.

For license servers, the syntax for the environment variable is: `@server_name`. For example, if the server machine is oak, set the environment variable to `@oak`.

The following sections explain how to set environment variables on the different platforms that MAK ONE products run on.

Windows

➤ **To set MAKLMGRD_LICENSE_FILE in Windows:**

1. On the Start menu, click the Settings button (⚙). The Settings window opens.
2. In the Search box, enter `environment`. A list of choices displays.
3. Select Edit the System Environment Variables. The System Properties dialog box opens to the Advanced tab.
4. Click the Environment Variables button. The Environment Variables dialog box opens.
5. Below the System Variables list, click New. The New System Variable dialog box opens.
6. In the Variable Name field, enter `MAKLMGRD_LICENSE_FILE`.
7. In the Variable Value field, enter `@server_name`, where *`server_name`* is the name of the license server.
8. Click OK to save out of each dialog box and set the variable.

Linux

On Linux, you set environment variables in your *.cshrc* (or equivalent startup file).

➤ To specify a license server, set the variable similarly to:

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

If you are using the sh or bash shells, you set environment variables in your *.profile* file (or *.bashrc*). Set the variable similarly to:

```
MAKLMGRD_LICENSE_FILE=@oak
export MAKLMGRD_LICENSE_FILE
```

Do not put spaces around the equal (=) sign.

You are ready to run the license server and use your new licenses or MAK ONE products.

2.4. Installing an RTI

An RTI is a software library (and perhaps supporting executables) that implements an HLA Interface Specification. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications need to use an RTI. MAK Technologies has the MAK RTI, but MAK ONE products do not require that you use the MAK RTI.

> Because of differences in the low-level network mechanisms used by different RTI implementations (which include, but are not limited to, packet layout), applications that want to interoperate in the same federation execution must use the same RTI implementation.

Because RTIs are usually provided as dynamic libraries that implement a fixed API, a federation can often switch from one RTI implementation to another between runs (without even recompiling the applications), but during each run, all participants must agree on which RTI to use, much as they must also agree on which FOM to use.

For the most recent information about the RTI versions supported by MAK ONE products, please visit the Product Specifications & Dependencies page (https://www.mak.com/mak-one-specs).

The following instructions explain how to install the MAK RTI. If you are using an RTI provided by another vendor, please refer to that RTI's documentation for installation and setup instructions.

2.4.1 Installing the MAK RTI

To install the MAK RTI, follow the instructions in Chapter 2 of *MAK RTI Users Guide*. To install another RTI, see the vendor's documentation.

Configuring Your System to Use the MAK RTI

The RTI dynamic libraries must be located somewhere on your dynamic library search path. The path is specified in the PATH environment variable. The path is indicated by an environment variable:

- Linux — LD_LIBRARY_PATH.
- Windows — PATH.
- The MAK RTI needs to know where to find these configuration files:
  - The federation configuration file (*.fed*, *.fdd*, or *.xml*) for your federation execution (required).
  - The RID file (*rid.mtl*) (optional).

Put the configuration files in the directory from which you are running, or set the environment variable RTI_CONFIG to the directory that contains them.

Running Applications with the MAK RTI

➤ **To run a MAK ONE product with the MAK RTI:**

1. Ensure that the license server is running.
2. Start the application. The RTI Assistant prompts you to choose an RTI configuration.
3. Choose a configuration. If necessary start the rtiexec.
4. Click Connect. The application should run.

In many cases, you do not need to run the rtiexec to use the MAK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MAK RTI.) For more information, please see your RTI documentation.

2.5. Using MAK ONE Products on Virtual Machines

You can run MAK ONE products on virtual machines. For more information, see the *Running MAK ONE Products on Virtual Machines or Docker Containers* guide on our website at https://www.mak.com/docs/products/MAK_ONE_Running_On_Virtual_Machines.pdf.

2.6. Localizing the Graphical User Interface

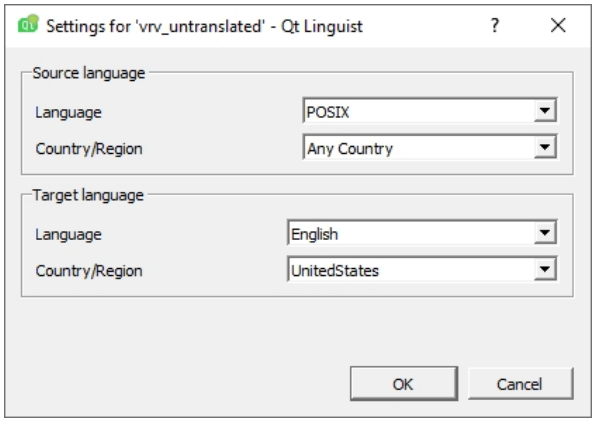
You can localize your MAK ONE product's graphical user interface so that it uses languages other than English. Your product includes a file called *./translations/product_untranslated.ts*, for example, *vrforces_untranslated.ts* or *vrv_untranslated.ts*. The file contains entries for all of the application-specific menu and dialog box text. The Qt Linguist tool, which is included with VR-Forces, lets you map the English text provided by MAK to your chosen language.

> The *nnn_untranslated.ts* file does not include all of the text that may be displayed in the GUI or in console and error messages.
>
> Some strings, such as those on the Create menu, are in configuration files. Others are in settings files. You can translate these strings by identifying the appropriate configuration file and translating the strings.
>
> VR-Forces has a utility that can scan system scripts and SMS files and create translation files for user-visible text. For details, see "2.6.2 Translating VR-Forces Scripts and Console Messages" on page 111.

➤ **To edit a resource file:**

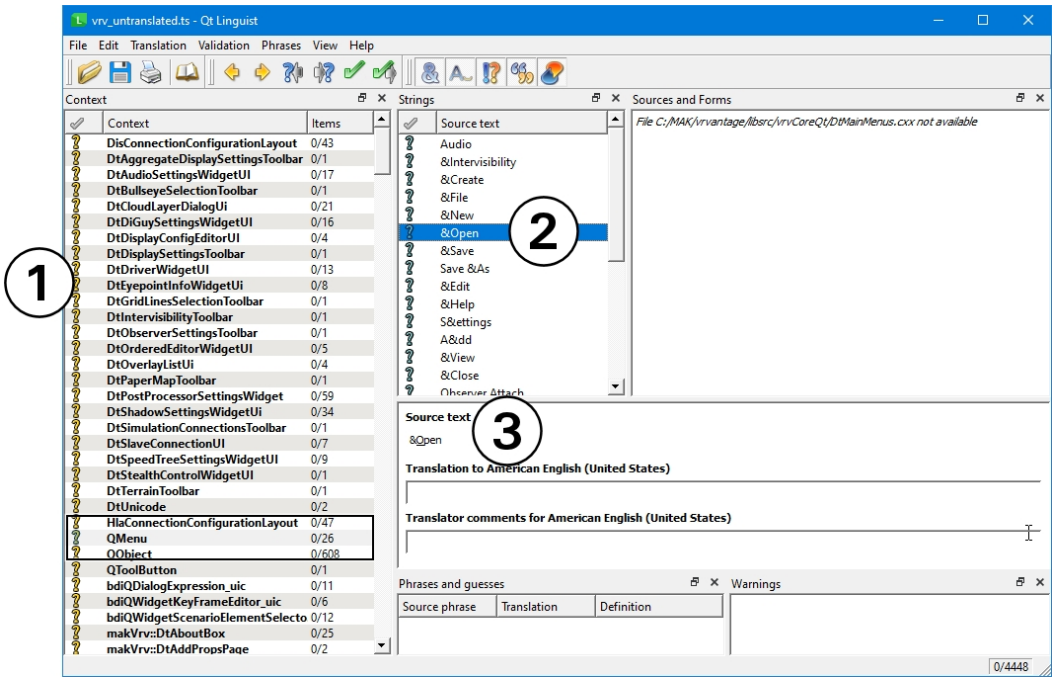
1. Start the Qt Linguist utility. It is *./translations/linguist.exe*
2. In the Qt Linguist window, choose **File > Open**. Select the file *./translations/product_untranslated.ts*. The Settings for *filename* dialog box opens (Figure 4).

Figure 4. Linguist settings dialog box



3. Select values for the source language, target language, and target region or country.
4. Click OK. The translation file opens and a list of classes displays on the left side of the window (callout 1 in Figure 5).

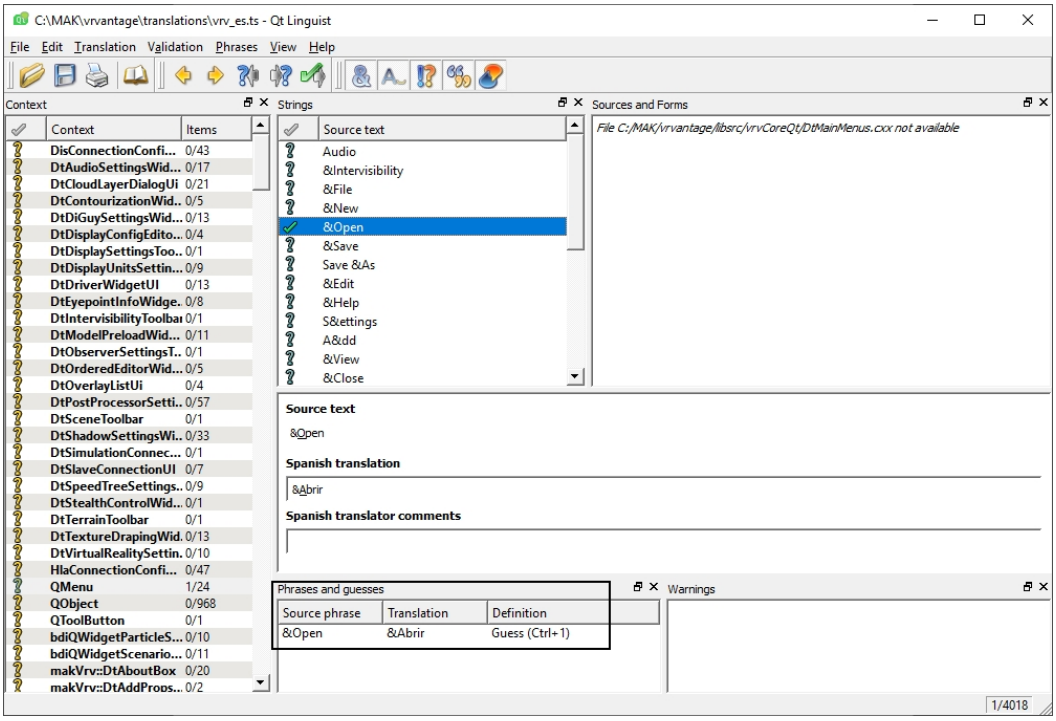
Figure 5. Linguist window



5. To translate text, select each class in turn and translate the text associated with that class. You do not have to understand anything about what the classes mean; you only need to translate the text. In Figure 5, **QMenu** is selected. The Strings window lists text that can be translated (callout 2). Text that has not been translated has a blue question mark in the checkmark column.
6. In the Source Text list, select the word you want to translate. The text displays under Source Text in the translation area (callout 3 in Figure 5).

7. Place the cursor in the text box below *language* Translation, where *language* is the target language that you selected when you opened the translation file.
8. Type the translation text.
9. When you are done and you are certain of the translation, select the Done and Next button () on the toolbar. A green check mark replaces the question mark (Figure 6). If you move to a different text string without clicking Done and Next, a yellow question mark displays for the string you translated. When you translate all elements for a class, a green check mark replaces the question mark next to the class name.

Figure 6. Completed translation for Open



10. Repeat this process for each element of source text for each class.
11. Choose **File > Release**. The file is saved in the format *filename.qm*.
12. Rename the file to a name that indicates the language to which you have translated the file, using the two-character abbreviation typically used with your language. For example, a Spanish translation would be renamed *product_es.qm*.
13. Save your translated *.ts* file.

2.6.1 Translating Other Interface Files

The *vrforces_untranslated.ts* or *vrv_untranslated.ts* file contains all the text used in the VR-Forces interface. However, there may be other text and messages that are not specific to VR-Forces that are generated by the software that implements the VR-Forces GUI. To translate this text, you need to translate the file *qt_untranslated.ts*. Follow the same procedure as you did for *vrforces_untranslated.ts*.

2.6.2 Translating VR-Forces Scripts and Console Messages

The *vrforces_untranslated.ts* file includes text for dialog boxes and interface elements created using the Qt Toolkit. However, there are many messages generated by code and the dialog boxes for scripts are generated programmatically, rather than through Qt ui files. Therefore they are not included in *vrforces_untranslated.ts*. VR-Forces has a utility that can scan all system scripts (those in a simulation model set) as well as some of the common SMS files, and create a TS file for each script and SMS. You can then translate the text in the individual TS files. This script also creates a TS file for DI-Guy files.

For this process to work, you must code messages in your scripts in a certain way. For details, please see "37.5. Coding Messages for Translation" on page 804. VR-Forces also has a way to code console messages so that they can be extracted and translated. For details, see the *VR-Forces Developers Guide*.

Generating TS Files for Scripts and Simulation Model Sets

If you code your scripts to support translation, you create TS files by running the translationFileCreate utility. By default, this application scans *./data/simulationModelSets* and its subdirectories for all system scripts and some generic files, and *./appData/settings* for DI-Guy files.

The utility creates an *untranslated.ts* file for each lua script. This file is saved in the directory that contains the script. It also creates a TS file for each simulation model set, which is saved in the SMS directory, and a *diguy_untranslated.ts* file, which is saved in *./translations*.

When you translate a script file, the translated file must remain with the sim engine(s) that are hosting the scripts. The rest of the translated files must exist on the user machine. The *.qm* file for DI-Guy and VR-Forces must live in the translations directory and the *.qm* file for the simulation model sets with the *.sms* files.

The syntax for translationFileCreate is:

```
translationFileCreate.exe [--directory filename] ...
[--smsfile string] ... [--nodiguy] [--nosms]
[--noscripts] [--] [-v] [-h]
```

Table 3. translationFileCreate arguments

Argument	Description
`--directory` *directory_name*	Directory to scan for files of any of the supported types (lua, sysdef, xml, mtl) as per creation rules (scripts, systems, supporting). Accepted multiple times.
`(--help | -h)`	Displays command-line usage information in a console window.
`--nodiguy`	Do not parse the DI-Guy configuration files.
`--noscripts`	Do not parse Lua files for translation.

Table 3. translationFileCreate arguments (continued)

Argument	Description
`--nosms`	Do not parse the simulation model set directories for their supporting translatable items.
`--smsfile` *`SMS_filename`*	Specific SMS file to use when translating simulation model set directories. Accepted multiple times. If not specified, processes all SMS files.
`(--version | -v)`	Displays version information.
`(--ignore_rest | --)`	Ignores the rest of the labeled arguments following this flag.

Generating TS Files for Object Console Messages

If the messages in your source code are coded for extraction to TS files, you can generate an *untranslated.ts* file for them. To generate TS files, run:

```
lupdate source_directory -ts my_new_translations.ts
```

The resulting TS file is saved to the location pointed to by the path given to `-ts`.

2.6.3 Applying the Language Files

➤ **To put the language file into effect, use one of the following methods:**

- Set your computer's environment to the proper language (LANG environment variable).
- Set the locale to the proper language.
- Start VR-Forces with the `(--locale | -G)` *`language`* command-line option, where *`language`* is the two-letter code for the local language.



3. VR-Forces Application Concepts

This chapter describes at a conceptual level the components of a VR-Forces simulation and how they work together.

3.1. The VR-Forces Program 114
3.2. GUI and Simulation Engine Concepts 114
3.2.1 How GUIs and Simulation Engines Work Together 114
3.2.2 How VR-Forces Simulation Engines Are Identified 115
3.2.3 VR-Forces Sessions 115
3.2.4 Coordinating Multiple GUIs 117
3.2.5 Working with Multiple Simulation Engines 118
3.3. Objects 119
3.3.1 The Object Parameter Database 119
3.3.2 Local Objects and Remote Objects 120
3.4. Representing and Managing Time in VR-Forces 120
3.4.1 Simulation Time 120
3.4.2 Time of Day 121
3.4.3 Exercise Clock Modes 122
3.5. Interactions 123
3.6. Advanced Terrain Navigation 123
3.7. Managing VR-Forces Settings 123
3.7.1 Synchronizing Settings Among VR-Forces Installations 126
3.7.2 Global Settings and Observer-Specific Settings 126

3.1. The VR-Forces Program

Unlike most computer applications, which have one executable file, VR-Forces has two executable files: a graphical user interface (GUI, also known as the front-end) and a simulation engine (sim engine or back-end). The GUI executable is vrfGui. The sim engine executable is vrfSim*protocol*, where *protocol* is DIS or HLA1516e.

By separating the simulation engine from the visualization program, VR-Forces allows you to run multiple GUIs, sim engines, or both. If you run multiple GUIs, several VR-Forces users can work together to create and manage scenarios. Using session IDs, you can specify which GUIs control which sim engines. The GUIs also display simulation objects received over the network from other simulations participating in an exercise. (You have a limited ability to interact with these remote objects. For more information, see "3.3.2 Local Objects and Remote Objects" on page 120, "43.10. Writing Plans for Remote Entities" on page 980, and "82.1. Attaching Components to Remote Entities" on page 1592.)

If you run multiple sim engines, the work of simulating simulation objects and tactical graphics can be shared by multiple computers, which should improve performance. If you save a scenario, VR-Forces records which objects are simulated by each sim engine, allowing the configuration to be reproduced the next time you run the scenario.



In this manual, most procedures apply to interaction with the GUI. For the sake of simplicity, unless we are discussing a subject that applies specifically to either the GUI or sim engine, we refer to VR-Forces in the sense of the combined interactions of both executables.

3.2. GUI and Simulation Engine Concepts

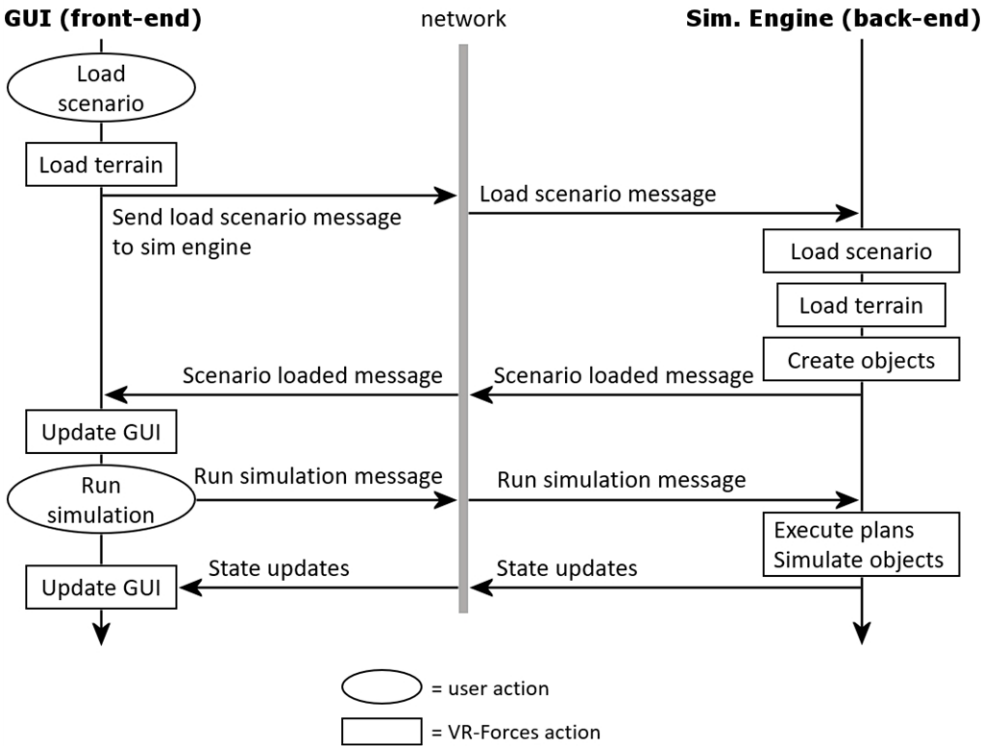
Although the separation of GUIs and sim engines—and the ability to run multiple instances of each—adds to the power and flexibility of VR-Forces, you need to understand how the various executables interact to ensure that your scenarios are created and run smoothly. In this section, we describe:

- How GUIs and sim engines work together.
- How sim engines are uniquely identified.
- Use of sessions.
- How to coordinate multiple GUIs.
- How scenarios are loaded and saved when you run multiple GUIs.

3.2.1 How GUIs and Simulation Engines Work Together

The VR-Forces GUI is an interface to the simulation engine. It communicates with the sim engine by sending messages over the network. (This relationship exists even if you are not connected to a network.) It displays the state updates and interactions that the sim engine generates and sends over the network. Figure 7 illustrates loading a scenario and running a simulation.

Figure 7. GUI and sim engine interaction



3.2.2 How VR-Forces Simulation Engines Are Identified

If you run multiple sim engines, you can specify which sim engine you want to use when you create simulation objects and tactical graphics. This feature allows you to spread the simulation load across multiple PCs. However, to do this, each sim engine must have a unique ID. VR-Forces sim engines are identified by their site ID and application number. Each sim engine must have a unique site:application identifier. You assign the site:application identifier when you start an application. It is up to you to make sure that there are no duplicate identifiers. (For details, see "4.1. Starting VR-Forces" on page 129.)

The preferred method for handling large scenarios is to optimize multiprocessor usage, as described in "Optimizing Multiprocessor Usage for Simulation Engine" on page 197. If you decide to run multiple sim engines, you should run them on separate machines.

> The site ID and application number concepts are borrowed from DIS. They also apply to VR-Forces when it is running in HLA.

3.2.3 VR-Forces Sessions

VR-Forces uses sessions to help manage the relationships between GUIs and sim engines. Sim engines are always part of a session. GUIs can join a session or operate independently of one. When a GUI is joined to a session, it can control the sim engines in that session. When a GUI is not joined to a session, it cannot control sim engines. However, it can open a terrain and view exercises running on its port. In this case, it is simply a viewer.

Use of sessions helps ensure consistency among GUIs that are controlling a common set of sim engines. The session mechanism ensures that:

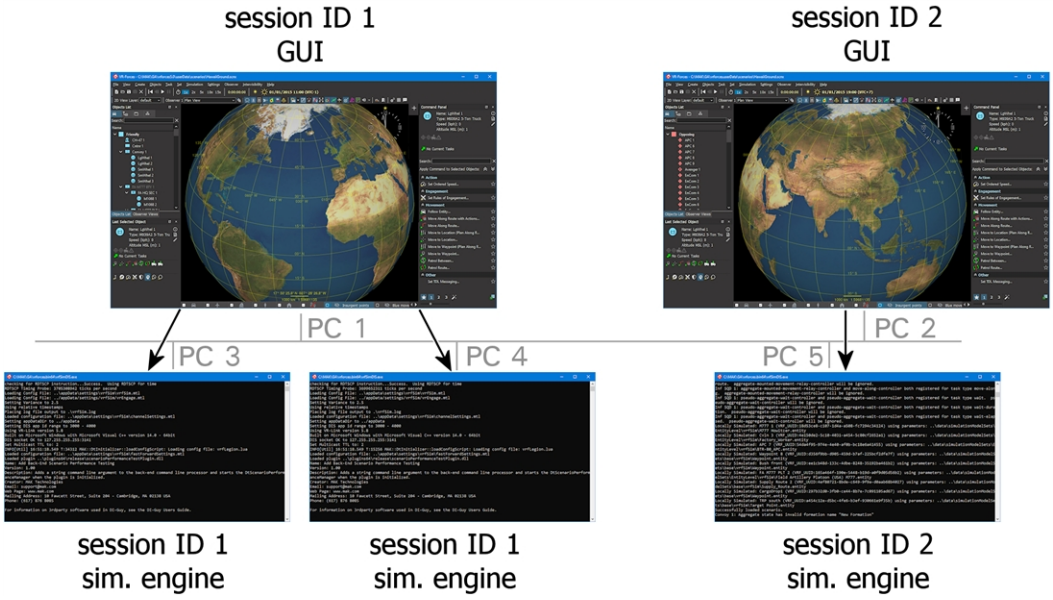
- All GUIs and sim engines in a session run the same scenario and use the same terrain.
- When a GUI joins a session that is running a scenario, it automatically loads the correct terrain for the scenario.
- If any GUI in a session tries to create a scenario or load a scenario, and a scenario is already running, the user is warned that this action will destroy the currently running scenario.
- If any GUI in a session loads a scenario or creates a new scenario, all of the sim engines and other GUIs in that session take the same action.

Sessions provide these additional benefits:

- They allow you to specify which GUIs control which sim engines. This feature could be useful, for example, if you want one set of VR-Forces applications to simulate friendly forces and another set to simulate opposing forces. If all applications were in the same session, participants could send orders to the simulation objects in the opposing force. By using separate sessions, this is not possible.

  In Figure 8, the GUI with session ID 1 can control the sim engines with that same session ID. The GUI with session ID 2 controls the sim engine with session ID 2.

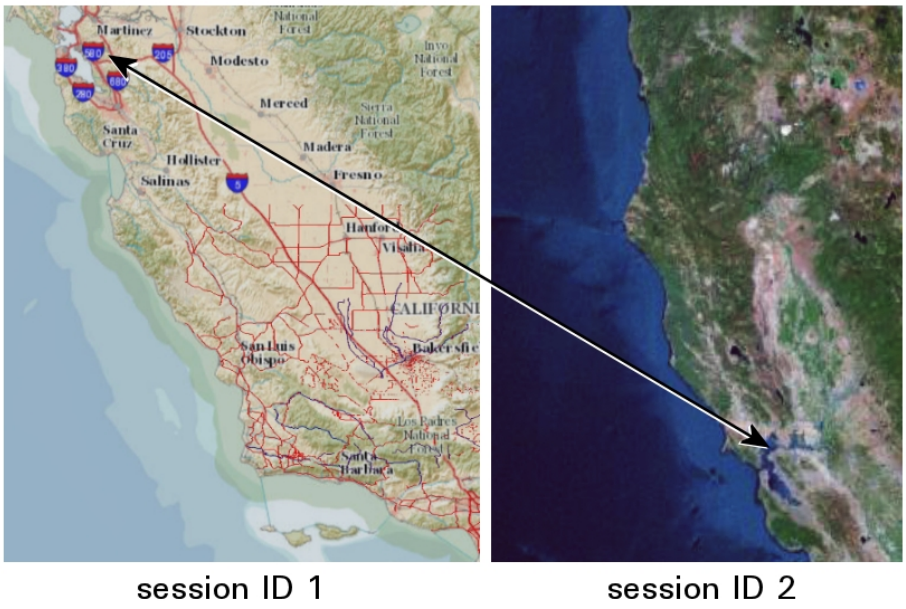
Figure 8. Using session ID to control sim engines



- Sessions allow you to load different terrain databases in different VR-Forces executables that are part of the same exercise. This feature may be helpful if you are participating in an exercise that covers a large geographical area, but only need to observe portions of that total terrain. Loading databases that cover a subset of the total exercise zone could improve the performance of your computer or the resolution of the terrain database.

In Figure 9, the two sessions have loaded databases that cover different (but overlapping) portions of California, USA. The entire exercise would cover the terrain covered by two databases.

Figure 9. Using session IDs to load different terrain databases



If two different sessions use terrain databases that overlap, note these possible sources of confusion:

- The GUIs in each session will see simulation objects from the other session that are simulated in the common portions of the terrain. However, the GUI for one session will have no ability to control the simulation objects that are simulated by the other session.
- If you save the scenario in one of these sessions, it only saves the simulation objects simulated in the session. It does not save the simulation objects from the other session, even though they may be visible in a GUI.
- If one of the sessions pauses or rewinds, the GUI in the other session will see the simulation objects in the overlapping region pause or return to their origin.

For more information, see "4.1.3 Specifying a Session ID" on page 133 and "4.3. Managing a GUI's Session Connection" on page 139.

3.2.4 Coordinating Multiple GUIs

Figure 7 on page 115 shows how a VR-Forces GUI sends messages to a sim engine and displays the simulation messages sent out by the sim engine. When you run VR-Forces with multiple GUIs, any of them can send control messages to the sim engine and all of them display the state updates. However, you would not want users at different GUIs to send conflicting or overlapping messages. For example, you would not want two different users to try to load the same scenario. Therefore, when you run multiple GUIs, bear in mind that:

- Only one GUI in each session can load a scenario or create a new one (they can all create the objects in the scenario).
- Any GUI can save a scenario. However, this could lead to multiple, unsynchronized versions of a scenario. It is probably best to reserve saving of scenarios to a specific GUI.
- Any object that you create in any of the GUIs is visible in the others and can be managed from them.
- You can close the scenario from any GUI.

VR-Forces provides some help managing multiple GUIs. For example, if one GUI loads a scenario, VR-Forces automatically asks the other GUIs to load the appropriate terrain. And if you close a scenario in one GUI, VR-Forces automatically prompts the other GUIs to close their terrains.

3.2.5 Working with Multiple Simulation Engines

As noted previously, you can distribute simulation of simulation objects among multiple sim engines across separate machines. The available sim engines are listed in the Selected Simulation Engine Toolbar (Figure 10). If you are running multiple sim engines, when you create simulation objects, you specify which sim engine you want to create them on by selecting a sim engine in the toolbar. (For more information, see "14.2. Selecting the Object to Create" on page 373.)

Figure 10. Selected Simulation Engine toolbar



When you run multiple sim engines, bear the in mind that:

- When you save a scenario created using multiple sim engines, the object map file ("12.4. The Object Map File" on page 357) records each object and the site and application ID of the sim engine on which it was simulated. When you want to replay that scenario, you should run a set of sim engines with the same site and application IDs. (If you do not, you can remap objects to the sim engines that you are running.)
- Multiple sim engines must all be started and ready when a scenario is loaded. There is no way for a required sim engine to join an exercise after it has started and catch up to the current state of the exercise.
- If, during an exercise, a sim engine drops out of the exercise, there is no good way for it to rejoin the exercise. If you load the scenario being run, it will be at the beginning, not at the point that the rest of the participants had reached.
- When you want a sim engine to join an exercise after it has started, you must explicitly load the terrain database using the **-T** command-line option. The sim engine will not receive any information about existing simulation objects and tactical graphics, but it can be used to create new simulation objects and tactical graphics going forward in the simulation.

- When you save a scenario that uses multiple sim engines, the final scenario files are composed of information generated by each sim engine and sent to the GUI, where it is combined. Therefore, if any of the sim engines has dropped out of the exercise or is otherwise unavailable, the scenario data will be incomplete.

Remapping Simulation Engines

The previous section lists the issues you should consider when creating, saving, and loading scenarios that use multiple sim engines. If you choose to run a scenario that requires multiple sim engines and some of the sim engines are not available, VR-Forces lets you remap objects from the missing sim engines to the available sim engines. You can remap sim engines automatically or manually.

If a scenario is missing its object map (*.omp*) file, you can remap the objects and create a new object map file. By default, in combined mode all the simulation objects are remapped to the combined mode sim engine. By default, in separate mode the remapping dialog box lists the missing sim engine as "UNKNOWN".

You might also want to manually remap sim engines to load balance a scenario. For more information, see "7.5. Load Balancing Scenarios" on page 255.

3.3. Objects

All simulation objects and tactical graphics are objects. The procedures for selecting, viewing, and organizing different kinds of objects are fairly similar. However since their purposes and use are quite different, for the most part we discuss simulation objects and tactical graphics separately.

3.3.1 The Object Parameter Database

VR-Forces creates objects based on parameter specifications in the object parameter database (OPD). In the OPD, object types are identified by an 8-field enumeration based on the SISO Enumeration Document (*Reference for Enumeration for Simulation Interoperability*). (The DIS enumerations have also been incorporated into the HLA RPR FOM.) In this manual, when we refer to entity or force type enumerations, we are referring to this set of enumerations. For information about the object parameter database and how to edit it, see the chapters in "Section XIII. Creating and Editing Simulation Models" on page 1301.

The objects in the OPD are organized into simulation model sets (SMS). An SMS contains all of the objects and supporting configuration files needed to populate a scenario.

VR-Forces includes these SMSs:

- *base.sms*. Included in all other SMSs for use in both types of scenarios.
- *EntityLevel.sms*, *MAKTest.sms*. Used by entity-level scenarios.
- *AggregateLevelBase.sms*. Included in aggregate-level SMSs.
- *AggregateLevel.sms*. Used primarily by legacy aggregate-level scenarios.

3.3.2 Local Objects and Remote Objects

From the point of view of a particular VR-Forces simulation engine, a local object is any object that is simulated in that sim engine. In that same sim engine, a remote object is an object that is received over the network from an external application, such as another VR-Forces sim engine, or a non-VR-Forces simulation. A given VR-Forces application participating in a networked exercise may know about a mixture of local and remote objects.

There are two categories of remote objects: objects simulated by VR-Forces applications, and objects simulated by applications other than VR-Forces. When VR-Forces learns about a non-VR-Forces remote object, it knows only what is transmitted in the standard DIS or HLA message. By contrast, VR-Forces remote objects share additional state information over the network, including a unique identifier (UUID), a string name, an echelon ID (if the simulation object is organized), and a string label.

> The VR-Forces GUI allows you to manage all the objects in a VR-Forces session regardless of which sim engine is simulating them. Therefore, in the interest of simplicity, we refer to simulation objects simulated by a VR-Forces sim engine, whether local or remote, simply as simulation objects and we refer to simulation objects simulated by applications other than VR-Forces as remote simulation objects.

You cannot control remote simulation objects from VR-Forces. However, you can attach VR-Forces components such as sensors and radios to remote simulation objects and you can write plans that use these components. For more information, see "43.10. Writing Plans for Remote Entities" on page 980 and "82.1. Attaching Components to Remote Entities" on page 1592.

3.4. Representing and Managing Time in VR-Forces

VR-Forces maintains an exercise clock. The clock tracks elapsed simulation time, date, and time of day. In HLA federations, it may be affected by federation time. These concepts also have a relationship to wall-clock, or real time.

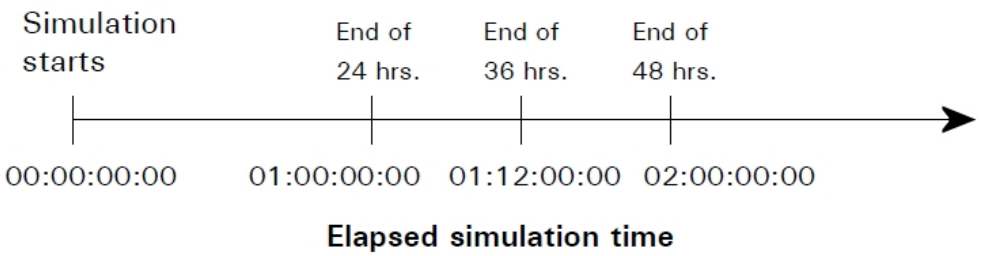
3.4.1 Simulation Time

Elapsed simulation time is the amount of time that has passed since the simulation started, and is expressed in days, hours, minutes, and seconds in the format 00:00:00:00.

Application developers can manipulate simulation time. Unless a local developer tells you otherwise, simulation time starts at 00:00:00:00.

Figure 11 illustrates the advance of simulation time.

Figure 11. Simulation time



You can view the simulation time using the Simulation Time Toolbar or the Simulation Time Panel. Both of these show the same time and can you can float or dock them. You can resize the Simulation Time Panel and the font size adjusts proportionately.

3.4.2 Time of Day

VR-Forces represents the day/night cycle with the Time of Day value. By default, it starts at 10:00 A.M in the time zone of the first simulation object created in a scenario, but can be set when you create a scenario. Time of day affects the simulation. For example, visual sensors are less able to detect objects at night than they are in the day.

Normally, time of day advances with simulation time, but you can change the time of day after the scenario is created, and even while the scenario is running. You can reset the time of day at any time through the environment settings, or through a plan with the Set Time of Day command. For details, see "11.1. Introduction to Environment Conditions" on page 324.

The display of time of day in the interface is 00:00 in hours and minutes. Time of day does not track number of days; when the time of day reaches midnight, it starts over at 00:00.

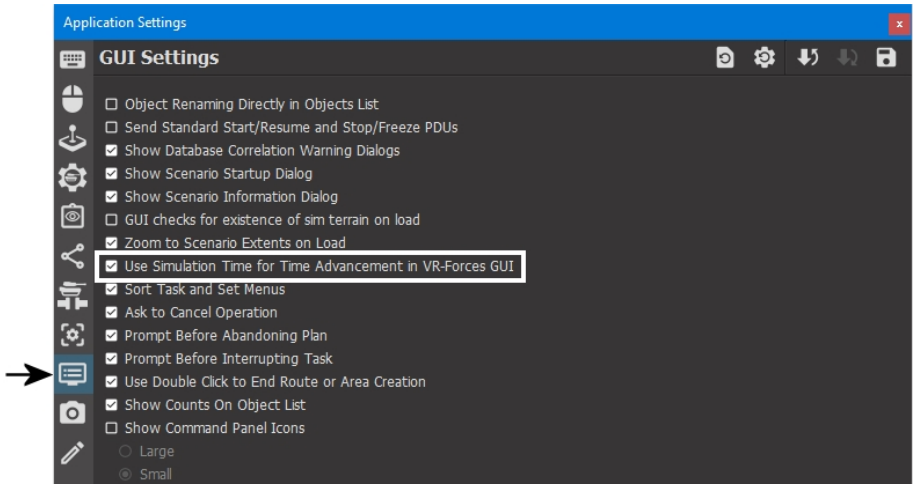
Advancing Time Using Simulation Time or Wall Clock Time

By default, time advances based on the simulation time. When VR-Forces uses simulation time for time advancement, simulation objects dead reckon correctly in the GUI when you run faster or slower than real time. (This also affects animations, such as helicopter rotors and ocean movement.) However, simulation objects published by non-VR-Forces federates will not dead reckon correctly. VR-Forces provides the option of advancing time based on wall clock time. In this mode, the time scale is not taken into account for dead reckoning.

➤ **To specify how time advances:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 12).

Figure 12. GUI Settings page



3. Select or clear the Use Simulation Time for Time Advancement in VR-Forces GUI check box.

3.4.3 Exercise Clock Modes

The exercise clock supports these modes:

- Variable-Frame Run-To-Complete mode advances simulation time by the amount of time passed since the last time the exercise clock was ticked. This mode is typical for distributed, interactive simulations. It does not provide repeatable results. Do not use this mode in time-managed HLA federations.
- Fixed-Frame Best-Effort mode advances simulation time by a fixed amount each frame, even if a frame takes longer than the fixed amount to compute. If the simulation takes less than the fixed frame time to compute, it waits until the frame time has elapsed before starting another.

  This mode is suitable for viewing or interactive use if your hardware can handle the frame rate. It is suitable for distributed use only in time-managed HLA federations.
- Fixed-Frame Run-To-Complete mode advances simulation time by a fixed amount each frame, even if a frame takes longer than the fixed amount to compute, just as in Fixed-Frame Best-Effort mode. However, if the simulation takes less then the fixed frame time to compute, it does not wait for the remainder of the frame time to elapse before starting the next frame.

  This mode is most useful for situations where you want a simulation to run with internal consistency and high fidelity, and want it to run to completion, but do not need to observe the simulation. So, for example, you might run a simulation overnight and view the results the following day. Fixed-Frame Run-To-Complete mode is not suited for interactive use. It is suitable for distributed use only in time-managed HLA federations. It disables the Simulation Time Scale Toolbar.

If you want to change the default clock mode, you must use the correct syntax. For information about how to set the exercise clock mode, see "Chapter 12. Scenario Files" on page 349.

Running a Simulation Faster than Real Time

VR-Forces supports simulation at rates faster than real-time by applying a multiplier to the tick time. For details, see "7.6.1 Changing the Simulation Speed" on page 260.

- Do not run your simulation at faster than real-time if you are interacting with other real-time simulations.
- Running a simulation faster than real-time can cause performance of simulation object models to degrade.

3.5. Interactions

VR-Forces displays interactions, such as detonations, that it receives in interaction messages or PDUs. You cannot select or otherwise manipulate them. You can only view them.

3.6. Advanced Terrain Navigation

VR-Forces supports advanced terrain navigation for entities based on the Autodesk® Gameware Navigation artificial intelligence software. Using Gameware Navigation Lab, you can create topological models of the virtual 3D environment that allow VR-Forces to compute, in real time, an optimized path for entity movement. As entities follow the path, they automatically avoid obstacles and other entities. The topological models are called navigation data.

Gameware Navigation Lab is included with VR-Forces. You can use it to view the navigation data included with VR-Forces terrain databases. If you want to generate navigation data, you must purchase an additional license.

For details about generating navigation data, see "Chapter 66. Generating Navigation Data" on page 1271.

3.7. Managing VR-Forces Settings

VR-Forces has the following types of settings:

- Independent, or display, settings are the individual settings managed on the Settings menu and the pages of the Display Settings dialog box. Changing an independent setting, such as enabling track histories or wireframe mode, does not affect other settings.

Independent settings are categorized as being global settings or observer-specific settings. Global settings apply regardless of the observer mode. Observer-specific settings can be different for each observer mode. For more information, see "3.7.2 Global Settings and Observer-Specific Settings" on page 126.

- Dependent settings are the schemas, model definitions, element definitions, and object type mappings that collectively determine how simulation objects and objects are modeled in the scene. Changing a dependent setting, such as a model definition may affect settings that depend on it, such as an entity definition.

VR-Forces manages settings:

- As you make changes to settings during a session, it saves the changes, except for model definitions, element definitions, and simulation object type mappings.
- It remembers the settings in effect when you exit and uses them as the default settings in your next session.
- During any particular session, you can revert to the default settings.
- You can save the current settings to a file.
- You can import saved settings.
- You can restore settings to those that were in effect when VR-Forces was first installed (factory settings).

Dependent settings have two additional options that independent settings do not have:

- You can save or restore some types of settings by specific setting.
- You can merge settings from a file into the current settings.

The procedures for saving and restoring settings are the same regardless of the types of settings. You start the process by clicking a button on a settings dialog box page. The way that VR-Forces saves and restores depends on the type of setting:

- When you initiate a save or restore operation on independent settings, it saves or restores all of the settings on the dialog box page from which you started the save or restore. If you have changed a setting on another page, it is not affected.
- When you initiate a restore operation on dependent settings, related settings may also be restored. VR-Forces displays a message listing the settings that will be restored.
- When you save dependent settings, you either export all settings of a given type or selected settings. For details, see the relevant chapters in *Visual Models in MAK ONE Applications* in the *Adding Content to MAK ONE Applications Reference Manual*.

Figure 13 illustrates the common settings management buttons. (The Visual Model Editors dialog box has some additional buttons.)

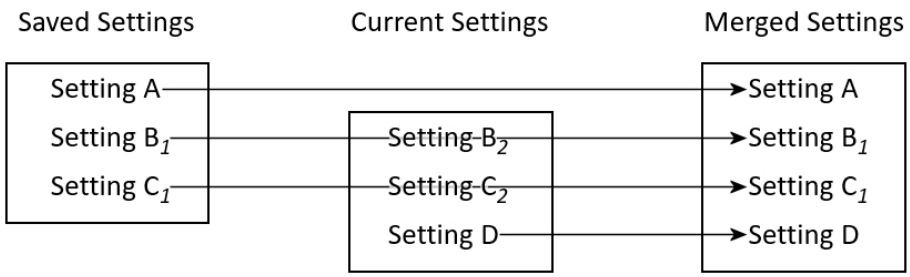
Figure 13. Settings buttons



- Revert to the default settings. Overrides any changes made to settings during this session.
- Restore the factory settings. Changes settings to the settings used when VR-Forces was installed. (The factory settings and default settings may be different.)
- Import settings from a file. Replaces the current settings with those stored in the selected file.
- Merge in settings from a file. When you merge settings from a file:
  - Saved settings that do not have a match in the current settings are added.
  - Saved settings replace matching settings.
  - Current settings that are not replaced (that is, for which there is no match in the saved settings) remain valid.

  Figure 14 illustrates how merged settings work. Setting A in the saved settings has no match in current settings, so it is added to the merged settings. Current settings B_2 and C_2 are replaced with saved settings B_1 and C_1. Current setting D has no match in saved settings, so it is retained in the merged settings.

Figure 14. Merged settings



- Save all settings to a file. Saves the current settings to the selected file.
- Export selected setting to a file.

- The buttons described in this topic are disabled on dialog box pages where their function is not supported. This includes most of the pages on the Terrain Settings dialog box.
- VR-Forces saves model definition and element definitions to *./SharedData/19/latest/ModelData/Configuration/Definitions*. Other settings are saved to *./appData/settings/vrfGui*.
- If you load an alternative settings file at startup, it is treated like an imported settings file.

3.7.1 Synchronizing Settings Among VR-Forces Installations

If your simulation is using several GUIs and sim engines on multiple computers, you may want to use the same settings across all of them, particularly if you have added model definitions and mappings to the factory settings. To synchronize settings, copy the data directories (*./appData*, *./data*, and *./userData*) from the computer on which you have updated them to the computers on which the other GUIs and sim engines are installed. Then import the settings into any Settings page that you customized on the source VR-Forces machine.

3.7.2 Global Settings and Observer-Specific Settings

VR-Forces has two types of independent settings – global settings and observer-specific settings.

Global settings affect the display regardless of which observer mode you are using. Global settings include simulation object labels, tactical graphics, and cockpit displays. The settings on the Settings menu are all global settings.

Observer-specific settings can be different for each observer mode. In fact, given a similar projection and model set, it is the different settings that distinguish one observer mode from another. For example, one difference between the defaults for Stealth observer mode and Out-the-Window observer mode is that Stealth mode has tactical smoke enabled and Out-the-Window mode has it disabled.



4. Running VR-Forces

This chapter explains how to start and stop VR-Forces and load terrain databases.

4.1. Starting VR-Forces 129
4.1.1 Starting VR-Forces from the VR-Forces Launcher 129
4.1.2 Starting Independent VR-Forces Executables 132
4.1.3 Specifying a Session ID 133
4.1.4 VR-Forces Startup Tutorial 134
4.2. The VR-Forces Window 135
4.2.1 Opening New Windows 137
4.2.2 Printing the VR-Forces Display 138
4.2.3 Taking Screen Captures 138
4.3. Managing a GUI's Session Connection 139
4.3.1 Configuring Session Messages and Join at Startup 140
4.3.2 Joining a Session 141
4.3.3 Resigning from a Session 142
4.4. Using HLA Time Management 142
4.4.1 VR-Forces Simulation Time Versus Federation Time 143
4.4.2 Configuring Time Management for HLA Exercises 143
4.5. Opening a Terrain Database 144
4.5.1 Loading a Terrain Database at Startup 145
4.5.2 Displaying Terrain Correlation Warnings 146
4.5.3 Closing a Terrain 146
4.6. Exiting VR-Forces 146
4.6.1 Disabling the Quit Prompt 146
4.7. Configuring Simulation Connections 147
4.7.1 Opening the Simulation Connections Configuration Dialog Box 147
4.7.2 Adding a Simulation Connection 148
4.7.3 Editing a Simulation Connection 149
4.7.4 Simulation Connection Parameters 150

4.7.5 Deleting a Simulation Connection 152
4.7.6 Configuring Auto Start 153
4.7.7 Displaying Connection Information 153
4.8. Configuring Applications to Launch 154
4.8.1 Adding an Application 154
4.8.2 Managing the Applications that Will Launch 155
4.8.3 Using Application Groups 156
4.9. Managing Plug-ins 156
4.9.1 Loading Plug-ins 157
4.9.2 Adding a Plug-in 158
4.9.3 Specifying the DLLs for a Plug-in 159
4.9.4 Adding a Plug-in Configuration 160
4.9.5 Deleting a Plug-in Configuration 161
4.9.6 Deleting a Plug-in 161
4.9.7 Viewing a List of Loaded Plug-ins 162
4.10. The VR-Forces Log Files 162

4.1. Starting VR-Forces

In "3.1. The VR-Forces Program" on page 114, we describe the VR-Forces GUI and sim engine applications. You can start GUIs and sim engines separately, which is called separate mode, or you can use the VR-Forces Launcher to start them together, which is called combined mode. The only real difference between the two startup modes is that in combined mode, when you shut down the GUI, the sim engine automatically shuts down also.

By default, all GUIs and sim engines start with session ID 1. GUIs automatically try to join a session. If no sim engines are running when a GUI starts, it does not join a session. However, if at a later time a sim engine with the same session ID starts, the GUI automatically joins the session.

> If you run VR-Forces for HLA, you must have installed an RTI and configured your system to use it. For details, see "2.4. Installing an RTI" on page 106.

For convenience, you can create batch files or scripts to start VR-Forces. See "Chapter 5. Command-Line Options" on page 163 for a list of command-line options.

4.1.1 Starting VR-Forces from the VR-Forces Launcher

You can run VR-Forces using DIS, HLA Evolved, or HLA 4.

For the HLA specifications, you can run the application using one of several different versions of the RPR FOM. With all these different possible startup configurations, particularly for HLA, it can be confusing to keep track of the options for starting VR-Forces. The VR-Forces Launcher simplifies the process of specifying the simulation standard to use when starting VR-Forces.

The VR-Forces Launcher allows you to start in either combined mode or separate mode.

➤ **To start VR-Forces using the VR-Forces Launcher:**

1. On the Start menu, choose **MAK Technologies > VR-Forces 5.2**.

   To start the Launcher on Linux (or from the Windows command line), in a console window, enter the command:

   ```
   vrfLauncher options
   ```

   For descriptions of the vrfLauncher command-line options, see "5.3. Command-Line Options for vrfLauncher" on page 187. The basic startup configurations for vrfLauncher are:

   — Start vrfGui and vrfSim (combined mode):

   ```
   vrfLauncher
   ```

   — Start in combined mode and do not display the Simulation Connections Configuration dialog box:

   ```
   vrfLauncher --connection connection_name
   ```

where *connection_name* is a previously defined configuration connection that includes at least one GUI and sim engine. See "5.3.1 Running in Combined Mode from the Command Line" on page 188 for details.

— Start vrfGui only:

```
vrfLauncher --gui
```

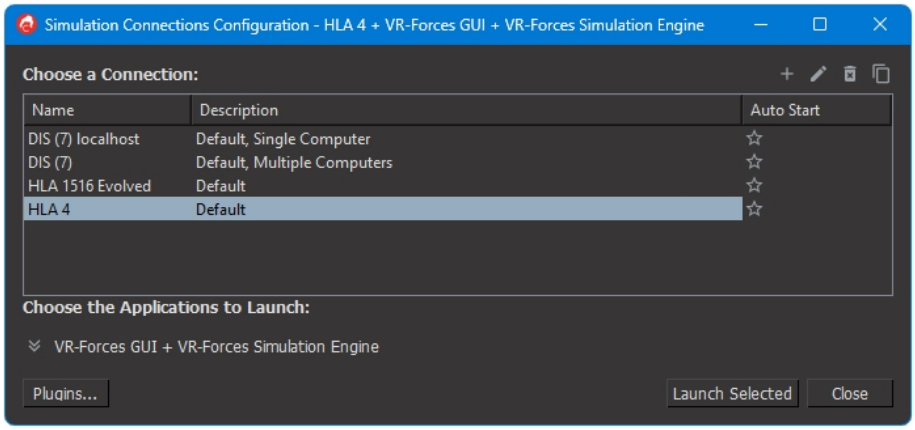
— Start vrfSim only:

```
vrfLauncher --sim
```

> Using `--gui` or `--sim` causes either the GUI or sim engine to start, respectively. If this is the first time that you are running VR-Forces, the VR-Forces Launcher opens and you can select the connection. After the first time, the GUI or sim engine starts using the most recent connection information.

2. If this is the first time that you are running VR-Forces, or if you have not specified a auto-start connection, the Simulation Connections Configuration dialog box opens (Figure 15).

Figure 15. Simulation Connections Configuration dialog box (showing advanced options)



3. In the Simulation Connections Configuration dialog box, select a connection configuration.

   You can create a new configuration using the Add Connection button (+) or edit an existing one using the Edit Connection button (✎). For details about creating and editing simulation connections, see "4.7. Configuring Simulation Connections" on page 147.)

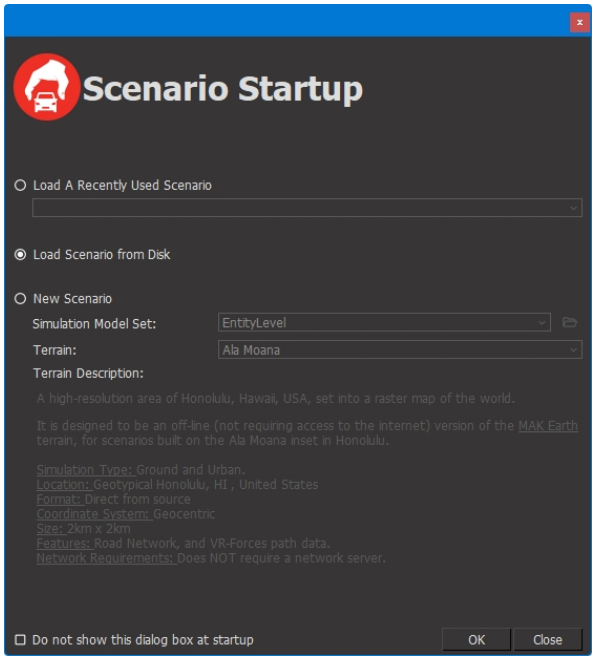
4. Optionally, click the arrows to expand the advanced options. You can specify what applications will run and configure how they connect to the simulation.

- To learn more about managing how the applications will behave when launched, see "4.8. Configuring Applications to Launch" on page 154.
- For lists of command-line arguments, see "Chapter 5. Command-Line Options" on page 163.

> You can look at an application's command-line arguments entries by clicking the View Arguments button ( ). When you click Launch Selected, these commands are executed. You can also copy the commands and use them in a batch file, script, or for debugging.

5. Click Launch Selected. The chosen VR-Forces executables start. When the VR-Forces applications finish their startup process, the Scenario Startup dialog box opens (Figure 16). Use it to quickly create or load a scenario using common options.

Figure 16. Scenario Startup dialog box



6. To create a scenario or load a scenario, select one of the options and then click OK. The options are:
   - New scenario. Select a terrain and simulation model set. VR-Forces creates a scenario using default scenario parameters. To specify scenario parameters when you create a new scenario, follow the procedure in "7.1. Creating a Scenario" on page 228.
   - Load a scenario. Choose a scenario from the recently used list or select Load Scenario From Disk. If you choose the latter, when you click OK, a file chooser dialog box opens. This option is the same as choosing **File > Load Scenario** in the GUI.
7. If you do not want any of the options, click Close. (This closes the Scenario Setup dialog box; it does not close VR-Forces.)

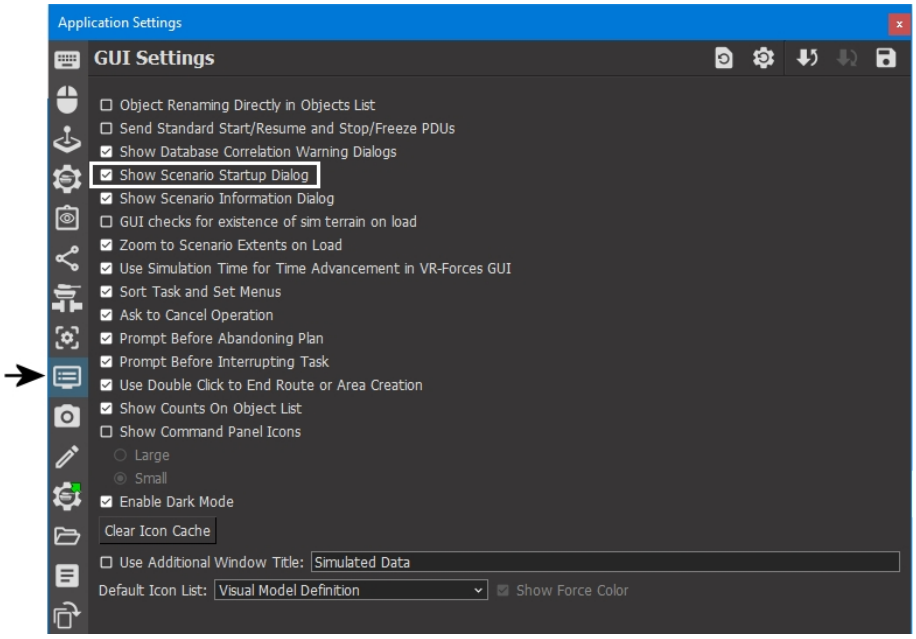
Enabling and Disabling the Scenario Startup Dialog Box

If you do not want VR-Forces to display the Scenario Startup dialog box every time it starts up, you can disable it. You can enable the dialog box at a later time.

➤ **To enable or disable the Scenario Startup dialog box:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 17).

Figure 17. GUI Settings page



3. Select or clear the Show Scenario Startup Dialog check box.

> You can also disable display of the Scenario Startup dialog box by selecting Do Not Show This Dialog At Startup on the dialog box itself.

4.1.2 Starting Independent VR-Forces Executables

You can start independent VR-Forces executables from the Launcher or the command line.

➤ **To start independent VR-Forces executables from the Launcher:**

1. On the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2**.

   To start the Launcher on Linux (or from the Windows command line), in a console window, enter the command:

   `vrfLauncher` *`options`*

   If this is the first time that you are running VR-Forces, or if you have not specified a auto-start connection, the Simulation Connections Configuration dialog box opens (Figure 15 on page 130).

2. Click the arrows to expand the advanced options. This displays the list of applications that you can run.
3. Optionally, edit the application you want to launch as described in "4.8. Configuring Applications to Launch" on page 154.
4. Click the Launch button (▶) at the far right of the application row.

   The application launches and the Simulation Connections Configuration dialog box remains open. The Next Application ID increments.

➤ **To start independent VR-Forces executables from the command line:**

1. Open a console window.
2. Change to the *./bin64* directory.
3. Enter a command for each VR-Forces executable you want to run. You must specify the protocol to use. For vrfSim, the protocol is part of the executable name. For vrfGui, it is a command-line argument. Specify a unique site:application identifier for each executable, for example:

   ```
   vrfGui --siteId 1 --appNumber 3000 {--dis |
       --hla1516e | --hla4} options
   vrfSimprotocol --siteId 1 --appNumber 3001 options
   ```

   where *protocol* is DIS, HLA1516e, or HLA4.

A VR-Forces window opens for each GUI you start and a console window opens for each sim engine. For an example of running multiple GUIs and sim engines, see "4.1.4 VR-Forces Startup Tutorial" on the next page.

- There is another launcher configuration available in *./factory/settings/vrfLauncher/Advanced*: HLA 1516 Evolved RPR 2.0 with MAK extensions and METOC.
- For more information on connecting and troubleshooting separate instances of MAK ONE products, please refer to the *MAK ONE Interoperability Guide*.

4.1.3 Specifying a Session ID

➤ To assign a session ID, specify it in the VR-Forces Launcher or use the `(--sessionId | -i) ID` command-line option.

```
vrfGui --siteId 1 --appNumber 3000 --sessionId 25 {--dis |
    --hla1516e | --hla4} options
vrfSimprotocol --siteId 1 --appNumber 3001 --sessionId 25 options
```

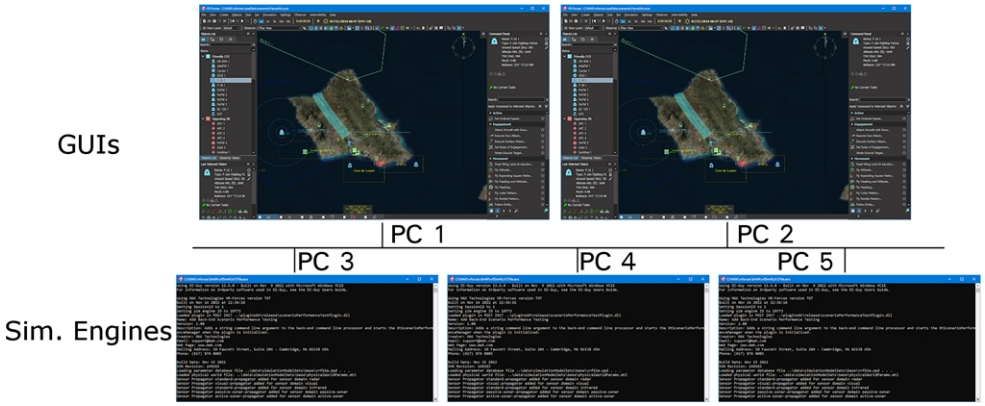
where *protocol* is DIS, HLA1516e, or HLA4.

The default session ID is 1. You cannot change the session ID during runtime.

4.1.4 VR-Forces Startup Tutorial

This section is a brief example of how to run multiple GUIs and sim engines on multiple computers. Examples are provided for DIS and HLA. For both protocols, assume the configuration in Figure 18. This configuration has two GUIs and three sim engines. Each is on a different computer. Assume that they are all running in the same session.

Figure 18. VR-Forces multicomputer configuration



As an alternative to starting individual GUIs and sim engines from the command line and specifying their protocol information in the command line, you can use vrfLauncher to start the GUIs and sim engines. Be sure to specify unique site and application numbers in vrfLauncher. For details about vrfLauncher command syntax, see "5.3. Command-Line Options for vrfLauncher" on page 187.

DIS Configuration

In a DIS exercise, you typically specify a port so that the exercise traffic is isolated from other network traffic. Each GUI and sim engine must have a unique site:application ID.

Run VR-Forces on each PC using these commands:

- PC 1:

  ```
  vrfGui --siteId 1 --appNumber 3000 --dis --disPort 1234
  ```

- PC 2:

  ```
  vrfGui --siteId 1 --appNumber 3001 --dis --disPort 1234
  ```

- PC 3:

  ```
  vrfSimDIS --siteId 1 --appNumber 3002 --disPort 1234
  ```

- PC 4:

  ```
  vrfSimDIS --siteId 1 --appNumber 3003 --disPort 1234
  ```

- PC 5:

  ```
  vrfSimDIS --siteId 1 --appNumber 3004 --disPort 1234
  ```

HLA Configuration

In an HLA exercise, you could simply specify the site:application ID and use default values to specify the HLA configuration parameters. Specification of the site:application ID is identical to that in DIS. For this example, we use HLA 1516 Evolved and specify a FOM Mapper other than the default.

Run VR-Forces on each PC using these commands:

- PC 1:

  ```
  vrfGui --siteId 1 --appNumber 3000 --hla1516e
  ```

- PC 2:

  ```
  vrfGui --siteId 1 --appNumber 3001 --hla1516e
  ```

- PC 3:

  ```
  vrfSimHLA1516e --siteId 1 --appNumber 3002
  ```

- PC 4:

  ```
  vrfSimHLA1516e --siteId 1 --appNumber 3003
  ```

- PC 5:

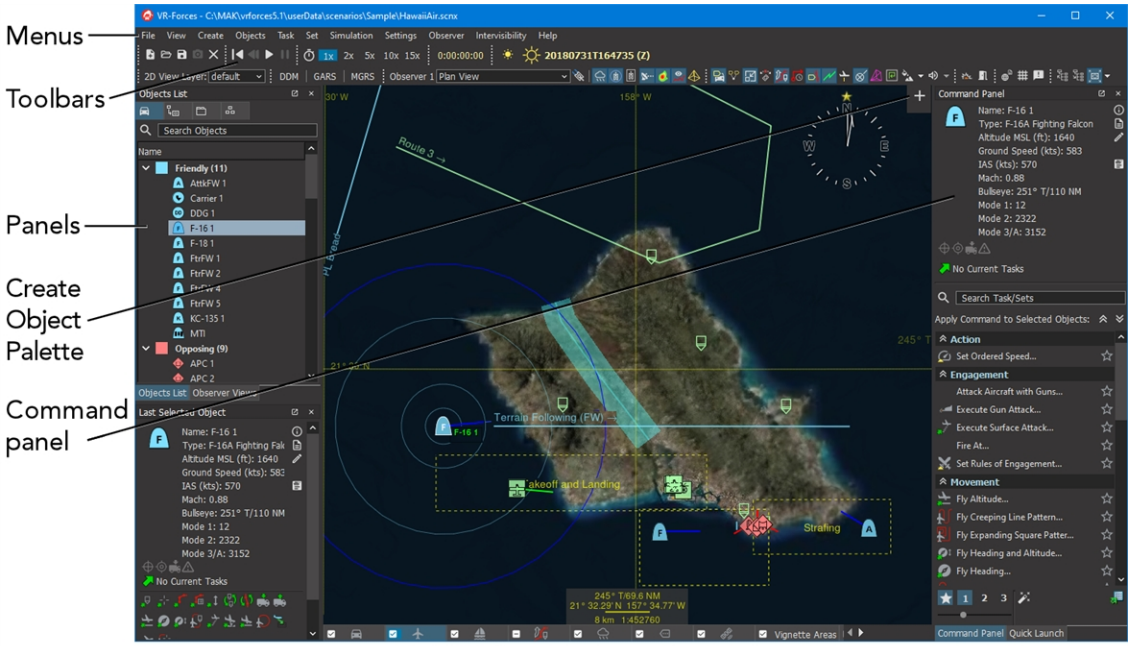
  ```
  vrfSimHLA1516e --siteId 1 --appNumber 3004
  ```

4.2. The VR-Forces Window

VR-Forces shows all the simulation objects and interactions on a terrain database. The VR-Forces window has these main components (Figure 19):

- Menus. Menus for accessing VR-Forces commands.
- Simulation control toolbars:
  - Simulation Control toolbar. Allows you to start, pause, and rewind a scenario.
  - Simulation Selection toolbar. Displays site and application numbers of multiple sim engines, if present.
  - Simulation Time toolbar. Displays the elapsed simulation time.
  - Simulation Time Scale toolbar. Allows you to control the speed at which the simulation runs.
- Application panels. Dockable panels that provide information and let you manage the observer and simulation objects.
- Convenience toolbars. Toolbars that let you quickly select frequently used menu commands and configuration settings.
- Object creation palettes. Palettes for creating simulation objects, tactical graphics, and props.

Figure 19. VR-Forces window (2D)



You can view the terrain in 2D (Figure 19), 3D (Figure 20), and exaggerated reality (XR) (Figure 21) modes.

Figure 20. VR-Forces window (3D)

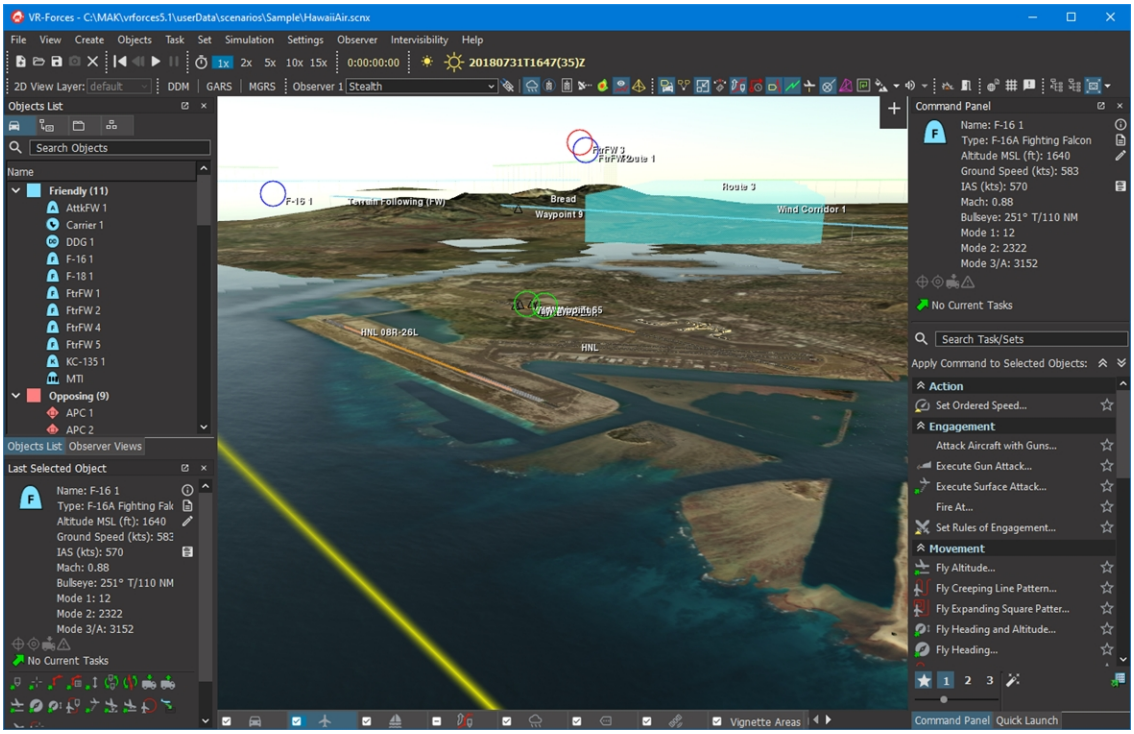
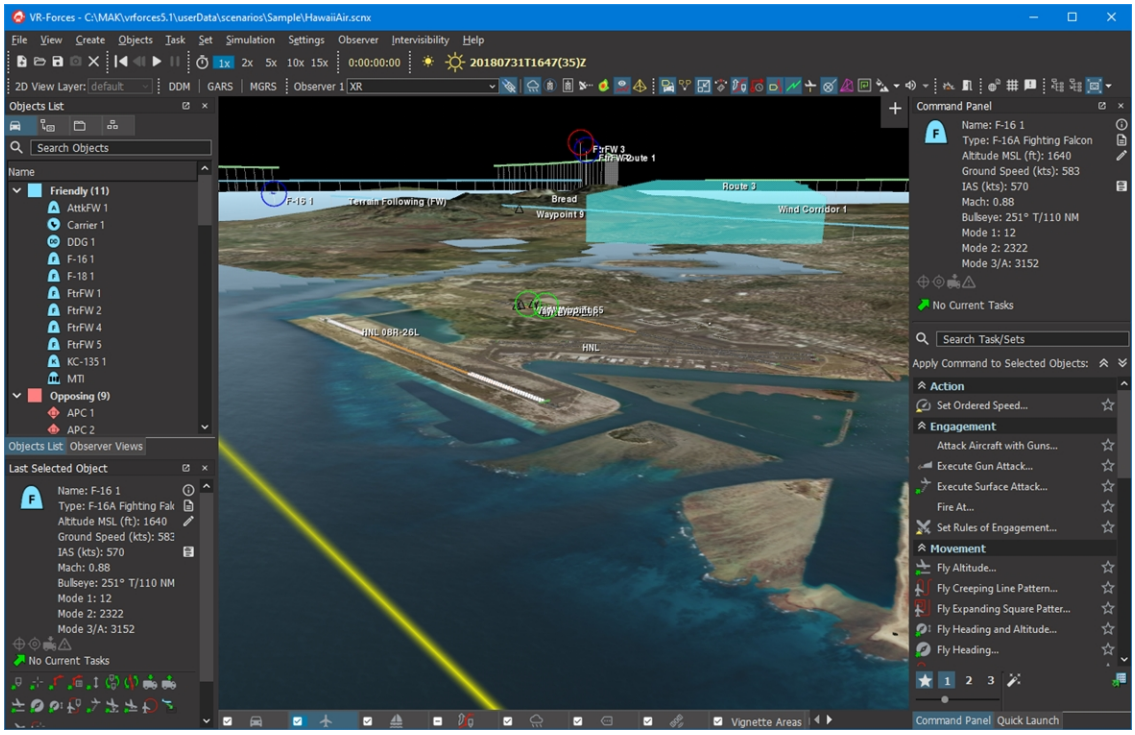


Figure 21. VR-Forces window (XR)



You can undock toolbars and panels and display them anyplace on your desktop. You can hide any toolbar that you do not want to be visible. (For details, see "Chapter 84. GUI Controls, Layouts, and Behaviors" on page 1621.)

4.2.1 Opening New Windows

In addition to the main VR-Forces window, you can open secondary windows. These windows do not have any of the menus or toolbars of the main window. However, you can navigate the windows using the same keyboard and mouse options as the main window. When a secondary window has focus, the settings on the Observer Settings toolbar apply to that secondary window and changes you make affect the window.

You can open windows with a 2D view or 3D view. Once a window is open, you can change the view by changing the observer mode.

Opening a secondary window is not the same thing as running a second instance of the GUI. When you close a GUI, all secondary windows associated with it are also closed.

➤ To open a new window, choose **File > New Stealth Window** or **File > New Plan View Window**.

You can also open a new window by opening an inset view. For details, see "62.1. Inset Views" on page 1224.

4.2.2 Printing the VR-Forces Display

You can print the VR-Forces display to any printer configured on your computer. None of the GUI controls or tabs print.

➤ **To print the VR-Forces display:**

1. Choose **File > Terrain > Print**. The Print dialog box opens.
2. Select a printer.
3. Click Print.

4.2.3 Taking Screen Captures

You can take screen captures of the VR-Forces window. None of the GUI controls or panels are saved.

By default, screen captures are saved to *./userData/screenshots*. You can change the default location. You can also change the key used for screen captures. For details about changing key mappings, see "86. Creating and Editing Key Mappings" on page 1649.

➤ To take a screen capture, press **Ctrl+F5**. It is saved to the default directory with a default name.

➤ **To take a screen capture:**

1. Choose **File > Terrain > Save Terrain as Image**. The Save Map Area dialog box opens.
2. Select the directory in which you want to save the image.
3. Type a name for the file.
4. Select the format you want to save it as.
5. Click Save.

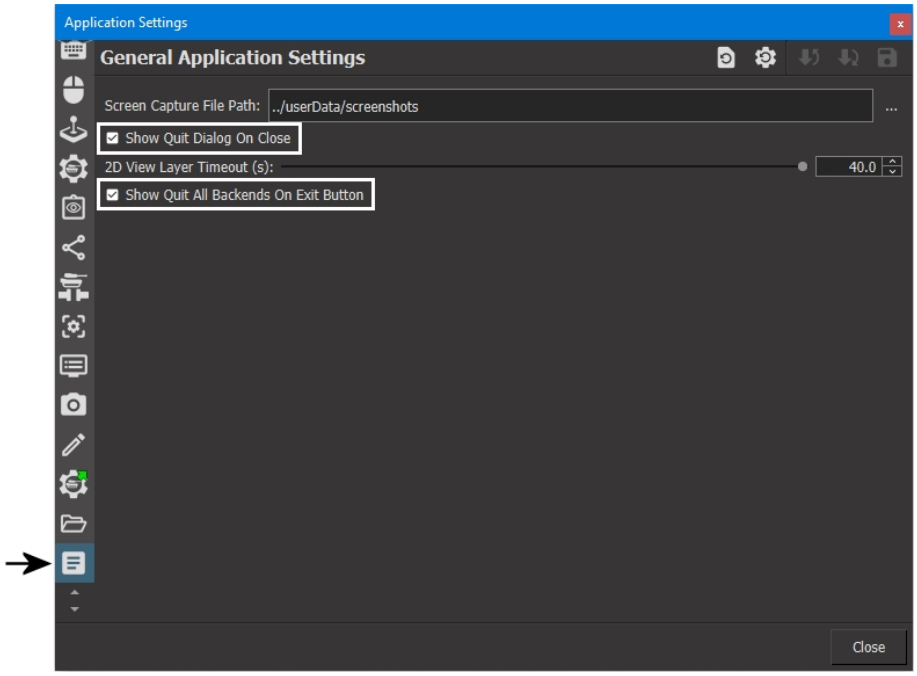
> On Linux machines, the screen capture the screenshot is not adjusted for gamma corrected. It appears dark in the default image viewer on Linux but displays correctly in other applications.

Specifying the Location of Screen Captures

➤ **To specify the location where screen captures are saved:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the General Application Settings page (Figure 22).

Figure 22. General Application Settings page



3. In the Screen Capture File Path box, specify the location where you want to save screen captures.

4.3. Managing a GUI's Session Connection

A GUI can be joined to a session or not joined. When a GUI is not joined to a session, it cannot create, load, or manage a scenario. It can only load a terrain and function as a viewer. You can tell a GUI's session status by the availability of Join Session and Resign Session commands on the File menu.

By default, when you start a GUI, it automatically tries to join a session. If there is no session running, it polls the network until one becomes available and then automatically joins it. Once a GUI joins to a session, you can manually resign from and rejoin it.

When a GUI joins a session, its behavior depends on the session settings set in the Application Settings dialog box, Session Settings page. The options are:

- Ask to join the session.
- Always join the session with the session database.
- Always join the session with whatever database is currently loaded.

⚠ If you do not load the terrain specified by the scenario, it is possible that visual and simulation terrains will not be synchronized.

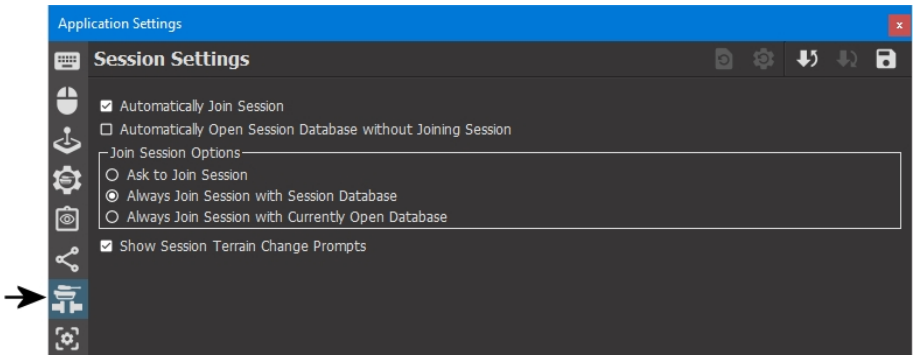
4.3.1 Configuring Session Messages and Join at Startup

You can configure the display of session-related messages and whether the GUI tries to join a session or open a terrain database automatically at startup.

➤ **To configure sessions:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Session Settings page (Figure 23).

Figure 23. Session Settings page



3. Select the options that you want to enable:
   - Automatically Join Session. If selected, when the GUI starts, automatically join a session if it is available. If cleared, the GUI does not join a session and does not prompt you to do so.
   - Automatically Open Session Database without Joining Session. This option lets VR-Forces automatically load the terrain database for the current session and put you in view only mode. You cannot create objects or control them. It interacts with Automatically Join Session as shown in Table 4.

Table 4. Join session and open session database interaction

Automatically Join Session	Open Session Database without Joining	Result
☑	☐	At startup, VR-Forces joins the session, loads the terrain, and you have full control of objects and scenario actions.
☐	☑	At startup, VR-Forces loads the terrain database. You have no ability to create published objects or control the scenario. (You can create local tactical graphics.) If you clear this check box after the terrain loads, it remains loaded.

Table 4. Join session and open session database interaction (continued)

Automatically Join Session	Open Session Database without Joining	Result
☑	☑	If you resign from a session, the terrain database stays open and you cannot create published objects or control the scenario. (You can create local tactical graphics.)
☐	☐	If, after starting VR-Forces, you select Automatically Open Session Database without Joining Session and a session is open, the terrain is loaded.

- Join Session Options. You can select one of these options for joining a session:
  - Ask to Join Session. Provides session data (Figure 24 on the next page) and prompts you to approve whenever you join a session.
  - Always Join Session with Session Database. Always join a session using the database in the loaded scenario. (Does not display a Join Session dialog box.)
  - Always Join Session with Open Database. Always joins a session with whichever terrain is currently open in the GUI. This might generate a terrain correlation warning.
- Show Session Terrain Change Prompts. Enable if you want to be prompted when the terrain in a session changes

4. Click Close.

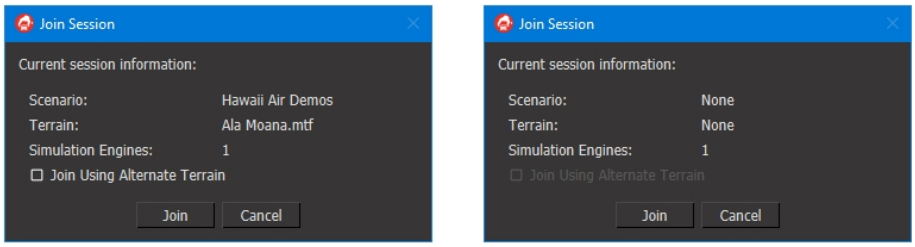
4.3.2 Joining a Session

When you join a session, all menu commands related to scenario management are enabled. If a scenario is loaded in the session, the terrain is automatically loaded in the joining GUI.

➤ **To join a session:**

1. Choose **File > Join Session**. If VR-Forces is configured to always join a session with the session database, the GUI loads the session database and joins the session. If VR-Forces is not configured to automatically use the session database, the Join Session dialog box opens (Figure 24). If a scenario is running, its details are listed.

Figure 24. Join Session dialog box



Scenario running — No scenario running

2. If a scenario is running, you can specify that you want to load an alternate terrain:
   a. Select the Join Using Alternate Terrain check box.
   b. Click Join. An Open Database dialog box opens.
   c. Select the terrain that you want to load.
   d. Click Open. The terrain is loaded and the GUI joins the session.

> ⚠ If you join a session using an alternate terrain, the GUI (visual) and sim engine (simulation) terrains might not be synchronized.

3. Click Join. The GUI joins the session.

4.3.3 Resigning from a Session

When you resign from a session, all menu commands related to scenario management are disabled. If you have a scenario loaded, the terrain is closed.

➤ To resign from a session, choose **File > Resign From Session**.

4.4. Using HLA Time Management

In HLA, use of time management allows the rate at which a federate proceeds to be controlled by another federate. Federates being controlled by other federates are said to be constrained. Federates that control other federates are time regulating. A federate can be both time regulating and constrained. It can also be unconstrained.

The main reasons why you might want to enable time management for VR-Forces are to run VR-Forces as part of a wider federation that is using time management or to synchronize multiple VR-Forces sim engines. Without time management, multiple sim engines can stay synchronized only if they all are able to maintain the same frame rate. However, if they cannot do so, they will no longer have a shared concept of simulation time and will become unsynchronized. By using time management services, multiple sim engines can be configured to run in lock-step with respect to simulation time, ensuring that events will always happen in the same order.

> ⓘ VR-Forces supports time management in the sim engine only.

For details about enabling time management, see "4.4.2 Configuring Time Management for HLA Exercises" below.

4.4.1 VR-Forces Simulation Time Versus Federation Time

Time management synchronizes federates to federation time. Federation time is the common model of time for the federation as whole. It is the time used in time stamp ordered messages (TSO), exchanged by time-managed federates. During the lifetime of a federation, federation time always increases. (There is no notion of rewinding back to an earlier time as with VR-Forces simulation time.)

VR-Forces simulation time is not the same as HLA federation time. It is the elapsed time within a scenario (the time displayed in the Simulation Time toolbar). When you first start a simulation, federation time and simulation time may be similar. However, if you rewind a time-managed simulation, the discrepancy between the two time streams will increase. This should not have any effect on running and rewinding scenarios. As long as all of the sim engines start from the same simulation state when you begin a simulation, they will be correctly synchronized (against federation time) as the simulation progresses.

4.4.2 Configuring Time Management for HLA Exercises

To use time management services in an HLA exercise, you must properly configure an RTI and enable time management in the VR-Forces sim engine.

Time management is enabled at the application level, not the scenario level. Scenarios created without use of time management work in time-managed mode. No time management-specific information is saved as part of a scenario.

If you enable time management in the sim engine, then whenever you create or load a scenario that specifies a fixed-frame mode (that is, the **frame-mode** parameter in the scenario file is set to either fixed-frame best effort or fixed-frame-run-to-complete), the sim engine will become a regulating and constrained federate. It will remain in this state until the scenario is closed.

A time-managed sim engine advances simulation time at a rate that is a function of the frame-mode specified in the scenario it is executing. If the **frame-mode** is fixed-frame, then it tries to advance simulation time at a rate approximating real time. However, if the sim engine has a heavy simulation load, or another time-constrained federate is advancing at a relatively slower rate, the sim engine may not achieve this rate. If the **frame-mode** is fixed-frame-run-to-complete, the sim engine tries to advance simulation time as fast as possible, independent of wall-clock time.

RTI Requirements for Time Management

To use time management in VR-Forces, your RTI must be properly configured to support this service. If you are using the MAK RTI, set the **RTI_useRtiExec** and **RTI_timeMgmt** parameters to 1. When you start the federation, you must run the rtiexec.

If you are using another RTI, please consult its documentation for configuration instructions.

Enabling Time Management for the Simulation Engine

➤ **To enable time management in the sim engine, use one of the following methods:**

- Set **runInTimeManagementMode** to 1 in *vrfSim.mtl*.
- Start the sim engine with the `--timeManagement` command-line option. For example, to start the sim engine in time management mode:

  `vrfSimHLA`*`protocol`* `--siteId 1 --appNumber 3001 --timeManagement`

- The sim engine will only run in time management mode (time regulating and constrained) while it has a scenario loaded.
- If a scenario is configured with **frame-mode** set to variable-frame, the sim engine will not become time regulating or constrained. It will print a warning to the console and run without time management enabled.

4.5. Opening a Terrain Database

If you are using multiple GUIs, when you load or create a scenario, the GUI where you issue the command loads the correct terrain database. The other GUIs prompt you to load the terrain. However, you can choose to load a different terrain. Also, if you want to use the GUI as a viewer, you can manually load a terrain database. (For a brief description of the databases supplied with VR-Forces, see "Terrain Databases Provided with VR-Forces" on page 1238.)

If you load a different terrain in the GUI than the one loaded in the sim engine, there might be correlation issues between the terrains. VR-Forces can warn you about correlation issues. For details, see "4.5.2 Displaying Terrain Correlation Warnings" on page 146.

You can only "open" terrains in MTF format. To open any other terrain format, you must compose a terrain, as described in the chapter *Composing Terrains*, or use an earth file, as described in the chapter *Paged and Streaming Terrains*, both in the *Adding Content to MAK ONE Applications Reference Manual*.

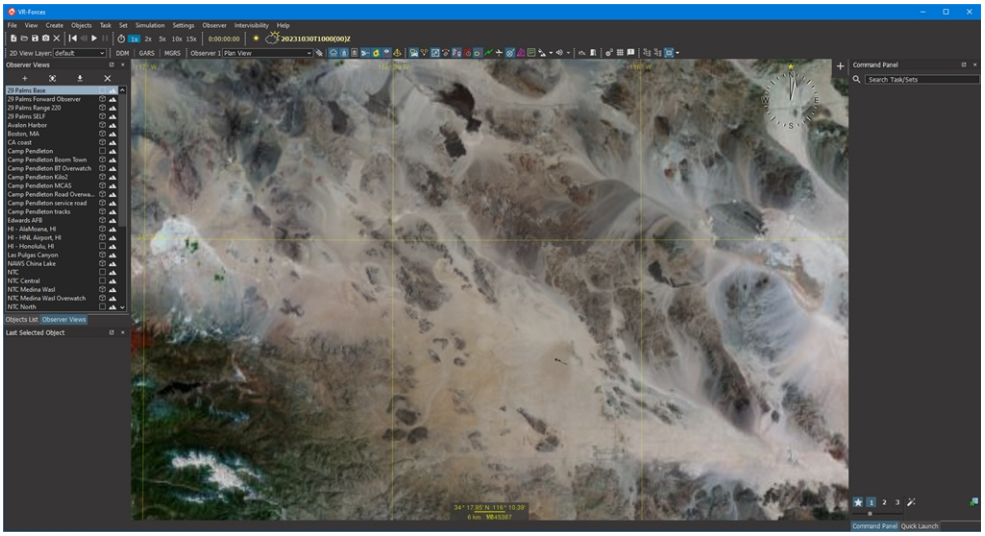
The first time you load a terrain (including the terrains supplied with VR-Forces), VR-Forces caches textures in its fast load format. This is a slow process. However, thereafter the terrain will load quickly. If you want to avoid runtime caching, you can preprocess the terrain with the medfTool before you load it. (For details about the medfTool, see "Compressing Model Files" in the *Adding Content to MAK ONE Applications Reference Manual*.) If you want to disable runtime caching, you can do so as described in "Configuring File Caching", also in *Adding Content to MAK ONE Applications Reference Manual*.

It is possible to open a terrain without having a scenario loaded. In such cases you can create tactical graphics and add props, but you cannot create simulation objects. To create and manipulate simulation objects, you must load or create a scenario.

➤ **To open a terrain database:**

1. In the VR-Forces window, choose **File > Terrain > Open Terrain**. The Open Terrain dialog box opens.
2. Select the terrain that you want to open.
3. Click Open. The database is displayed in the VR-Forces window (Figure 25).

Figure 25. VR-Forces window with a database loaded



- If you load a terrain database while one is currently loaded, VR-Forces closes the current database and then opens the new one.
- Paged terrains might not be visible when loaded in Plan View mode. To see the terrain, press **E** to zoom in. This problem potentially exists for all paged terrains whose page-in distance is less than the default observer distance in Plan View mode.
- Some MetaFlight terrains do not display in the 2D view when first loaded. To see the terrain, change to a 3D view mode, and press the space bar (center view on terrain) to move the eyepoint to the terrain location. After this, the terrain can be viewed in the 2D view.

4.5.1 Loading a Terrain Database at Startup

You can load a terrain database at startup by specifying it on the command line. If you start VR-Forces while a session is running, it can load the appropriate terrain database automatically. For details, see "4.3.1 Configuring Session Messages and Join at Startup" on page 140.

➤ To load a terrain database when you start VR-Forces, use the (`--terrainDatabase` | `-T`) *`terrain_database`* command line option.

```
vrfSimprotocol --siteId 1 --appNumber 3001 --terrainDatabase
  "../userData/terrains/myDatabase.mtf"
```

where *`protocol`* is `DIS`, `HLA1516e`, or `HLA4`.

4.5.2 Displaying Terrain Correlation Warnings

By default, VR-Forces displays a warning message if the terrain loaded in the GUI might not correlate well with the terrain loaded in the sim engine. You can enable and disable this warning message.

➤ **To enable or disable terrain correlation warning messages:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page.
3. Select or clear the Show Database Correlation Warning Dialogs check box.
4. Click Close.

4.5.3 Closing a Terrain

➤ **To close a terrain, use one of the following methods:**

- Choose **File > Terrain > Close Terrain**.
- Choose **File > Terrain > New Terrain**.
- Open a different terrain.

4.6. Exiting VR-Forces

When you exit VR-Forces from the GUI, all GUI windows close. In combined mode, the simulation engine that started with the GUI also closes. If you are not running in combined mode or have multiple simulation engines, you must also close all simulation engines (either manually or using the correct Exit option). There are two Yes options on the exit prompt:

- Yes. In independent mode, closes the GUI. In combined mode closes the GUI and the simulation engine that was started with the GUI.
- Yes, and Quit All Sim Engines. Closes the GUI and all simulation engines in the session, regardless of how you started them, which computer is running them, and which operating system they are running on.

You can disable the exit prompt if you want to. In that case exiting acts as if you clicked Yes on the exit prompt. For details, see "4.6.1 Disabling the Quit Prompt" below.

➤ **To exit the VR-Forces GUI:**

1. Choose **File > Exit**. An exit prompt opens.
2. Click the button for the exit option you want.

➤ To exit a simulation engine from the console window, press **Q** and then **Enter**.

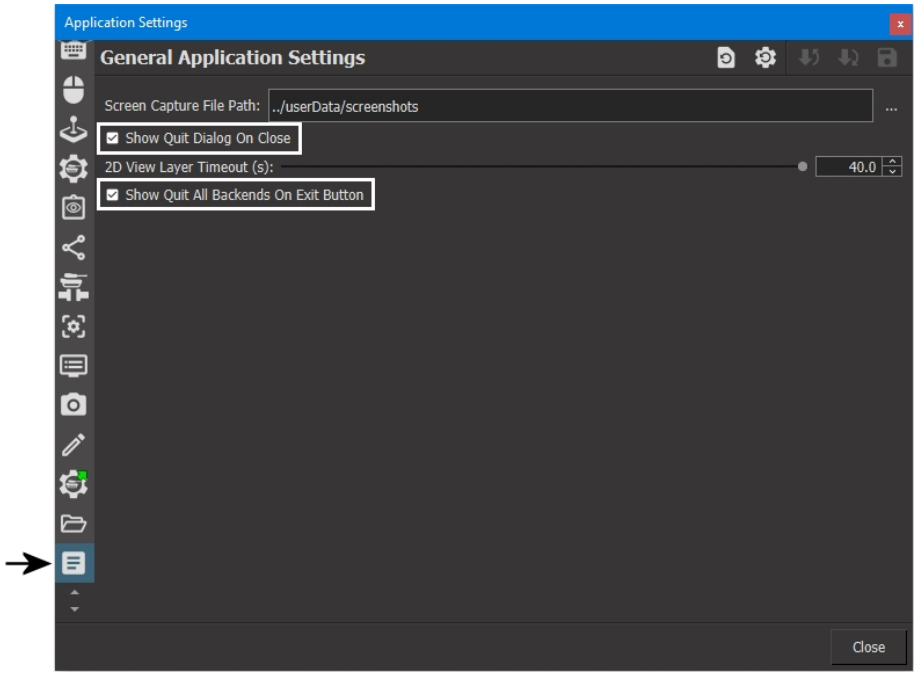
4.6.1 Disabling the Quit Prompt

When you exit VR-Forces, you are prompted to confirm that you want to exit. You can disable this prompt.

➤ **To disable the quit prompt:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the General Application Settings page (Figure 26).

Figure 26. General Application Settings page



3. Clear the Show Quit Dialog On Close check box.

4.7. Configuring Simulation Connections

A simulation connection specifies the connection parameters for a DIS or HLA simulation connection. VR-Forces comes with these connection configurations:

- DIS (7) localhost (for use when disconnected from a network or if you do not want to interact with network traffic).
- DIS (7).
- HLA 1516 Evolved.
- HLA 4.

You can edit the supplied configurations or add your own connection configurations that use, for example, a different DIS port or your own FOM Mapper.

4.7.1 Opening the Simulation Connections Configuration Dialog Box

Simulation connection configurations are maintained in the Simulation Connections Configuration dialog box. This dialog box opens automatically when you start VR-Forces or run the vrfLauncher, unless you choose a configuration as the default connection for auto connection. If Auto Connect is enabled and you want to change the connection, you can open the dialog box from the Tools menu.

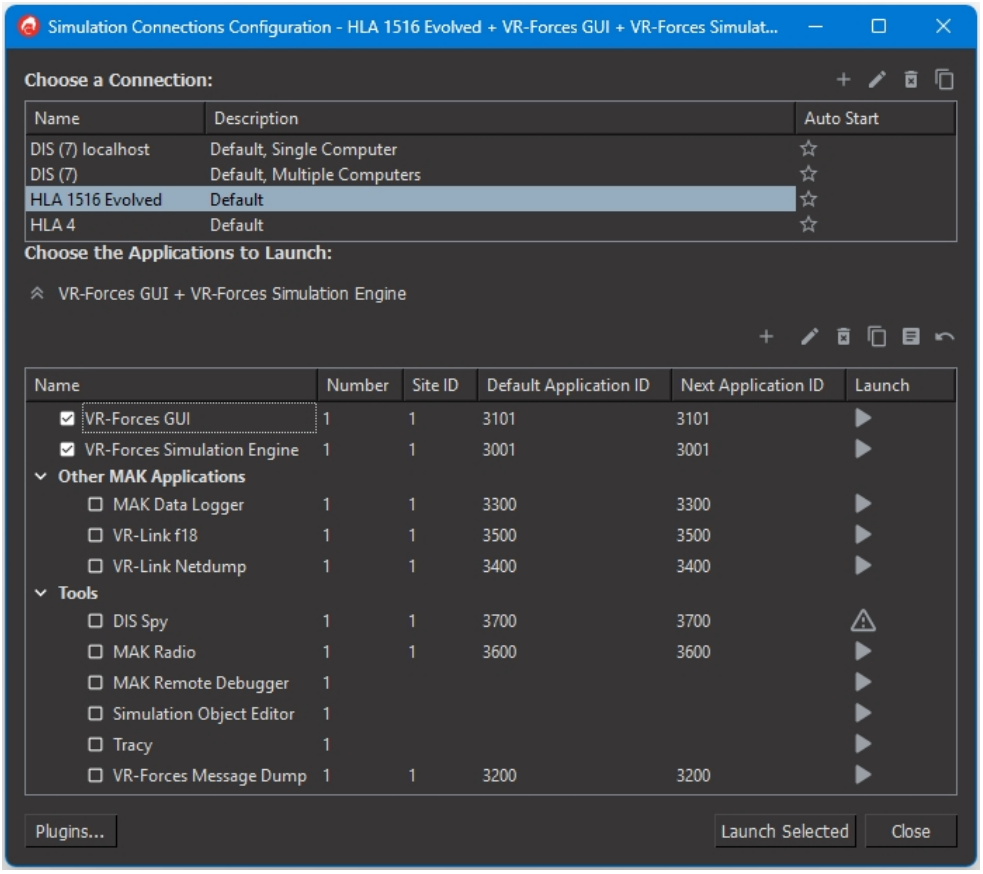
➤ To open the Simulation Connections Configuration dialog box from the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut.

➤ To start the vrfLauncher on Linux (or from the Windows command line), in a console window, enter:

```
vrfLauncher options
```

For descriptions of the vrfLauncher command-line options, see "5.3. Command-Line Options for vrfLauncher" on page 187.

Figure 27. Simulation Connections Configuration dialog box



4.7.2 Adding a Simulation Connection

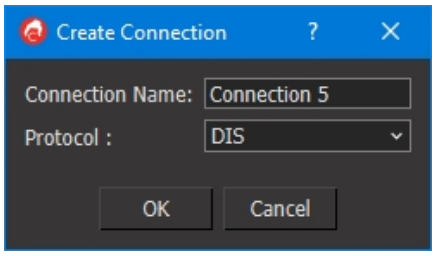
When you add a new simulation connection, it has the default values for the protocol that you select. You can then customize it to meet your needs. You can also add new connections by copying an existing connection.

➤ **To add a simulation connection:**

1. On the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens (Figure 27 above).

2. Click the Add Connection button (+). The Create Connection dialog box opens (Figure 28).

Figure 28. Create Connection dialog box



3. In the Connection Name box, type a name for the connection.
4. Select a protocol from the Protocol list.
5. Click OK. The Add Connection dialog box opens.
6. Edit the parameters as desired. They are described in "4.7.4 Simulation Connection Parameters" on the next page.

Copying a Simulation Connection

You can create a new simulation connection by copying an existing connection.

➤ **To copy a simulation connection:**

1. On the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens (Figure 27 on the previous page).
2. Click the Copy Connection button (). The Copy Connection dialog box opens.
3. Accept the default name or type a new name.
4. Click OK. The new connection is added to the list.
5. Click the Edit button (). The Edit Connection dialog box opens.
6. Edit the parameters as desired. They are described in "4.7.4 Simulation Connection Parameters" on the next page.

4.7.3 Editing a Simulation Connection

➤ **To edit a simulation connection:**

1. On the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens (Figure 27 on the previous page).
2. In the list of connections, select the connection that you want to edit and then click the Edit button (). The Edit Connection dialog box opens.

3. Edit the parameters as desired. They are described in "4.7.4 Simulation Connection Parameters" below.

You can also edit connections from the VR-Forces Launcher.

4.7.4 Simulation Connection Parameters

The HLA and DIS simulation connection parameters are equivalent to command-line options for DIS and HLA. Most are defined in the exercise connection configuration (exConnConfig) file, while some are also available in the GUI.

Each connection has a Name and Description, as well as parameters for either the HLA or DIS connection. Table 5 describes the Name and Description, Table 6 describes the connection parameters for HLA, and Table 7 describes the connection parameters for DIS. The latter two tables cross-reference the parameters to the command-line options in "Chapter 5. Command-Line Options" on page 163.

By default, VR-Forces uses the configuration file *./appData/settings/connections/MAK-ONE-YYYY-Config.xml*. You can open or copy a configuration file from the Add or Edit Connection dialog box. The relevant configuration file parameters are listed below in the tables for the HLA and DIS connections. For more information about the configuration file, see the *MAK ONE Products Interoperability Guide*.

Table 5. Connection configuration parameters for all connections

Parameter	Description
Name	The name that displays in the VR-Forces Launcher. You can use this in conjunction with the `(--connection | -c) connection_name` command-line option where *`connection_name`* is the Name. If there are spaces in the Name, you must surround it with quotes, for example, `"DIS (7)"` or `"HLA 4"`.
Description	The description that displays in the VR-Forces Launcher.

Table 6. Connection configuration parameters for HLA

Parameter	Command-line equivalent	Description
GUI		
Network Interface Address	`--deviceAddress` *`address`*	The IP address of the host computer.
Session ID	`(--sessionId | -i)` *`ID`*	Specifies the session ID for this instance of the GUI.
Use Absolute Time Stamps	`--useAbsoluteTimeStamps`	Enables use of absolute timestamps. Otherwise uses relative timestamps.

Table 6. Connection configuration parameters for HLA (continued)

Parameter	Command-line equivalent	Description
⚠ If you select `--useAbsoluteTimeStamps`, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.		
Configuration File		
Federation Name	`(--execName | -x)` *`exec_name`*	Specifies the name of the federation execution. See "5.5.1 Specifying the Federation Execution" on page 190.
FED File Name	`(--fedFileName | -F)` *`fedfile_name`*	Specifies the FED file to use. See "5.5.2 Specifying the FED File" on page 191.
RPR FOM Version	`--rprFomVersion` *`version_number`*	See "5.5.3 Specifying the RPR FOM Version" on page 191.
RPR FOM Revision	`--rprFomRevision` *`revision_number`*	RPR FOM revision for RPR FOM 2.0, draft 17.
NETN FOM Version	`--netnFomVersion` *`version_number`*	Specifies the NETN FOM version as a double. The default (and currently the only supported version) is 3.0.
NETN FOM Revision	`--netnFomRevision` *`revision_number`*	Specifies the internal revision of the implementation of the NETN FOM version as an integer. The default (and currently the only supported revision) is 1.
FOM Modules	`--setFomModuleList`	Specifies one or more HLA Evolved or HLA 4 FOM modules. It can take multiple entries, which you should list in their hierarchical order to prevent any issues related to dependencies.

Table 7. Connection configuration parameters for DIS

Parameter	Command-line equivalent	Description
GUI		
Network Interface Address	`--deviceAddress` *`address`*	Specifies the address of the network card to use for UDP traffic.

Table 7. Connection configuration parameters for DIS (continued)

Parameter	Command-line equivalent	Description
Destination Address	`(--destAddrString | -A)` *address*	Specifies a specific destination address (point-to-point). See "5.6.2 Specifying Point-to-Point or Multicast Operation" on page 192.
Port	`(--disPort | -P)` *port*	Specifies the DIS port. In the GUI, you can enter a port or specify that the application use the value in the configuration file. See "5.6.1 Specifying the Port Number" on page 192.
Session ID	`(--sessionId | -i)` *ID*	Specifies the session ID for this instance of the GUI.
Use Absolute Time Stamps	`--useAbsoluteTimeStamps`	Enables use of absolute timestamps. Otherwise uses relative timestamps.
⚠ If you select `--useAbsoluteTimeStamps`, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.		
Configuration File		
Exercise ID	`(--exerciseId | -x)` *ID*	Specifies the DIS exercise ID. See "5.6.5 Specifying the Exercise ID" on page 193.
DIS Version	`--disVersion` *version*	Specifies the DIS version. 7 is currently the only valid setting. See "5.6.6 Specifying the DIS Version" on page 193.
Use IPV6	`--useIpv6`	Specifies that the DIS exercise connection use the IPV6 protocol, if available.

4.7.5 Deleting a Simulation Connection

➤ **To delete a simulation connection:**

1. On the Windows Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens (4.7.2 on page 148).
2. In the list of connections, select the connection that you want to delete.
3. Click the Delete button (). A confirmation prompt opens.
4. Click Yes.

4.7.6 Configuring Auto Start

If you know that you will usually or always want to use the same simulation connection, you can set up Auto Start and you will not be prompted to select a connection every time you start VR-Forces. You can select only one connection configuration.

➤ **To set up Auto Start:**

1. You can configure auto start in either the Simulation Connections Configuration dialog box or the Configure VR-Forces Connections tool.
   - In the Simulation Connections Configuration dialog box: Choose **MAK Technologies > VR-Forces 5.2**.
   - In the connections configuration tool: Choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens (Figure 27 on page 148).
2. In the list of connections, next to the connection that you want to use every time you start VR-Forces, click the star in the Auto Start column (☆) to select it.

Disabling Auto Start

➤ **To disable Auto Start:**

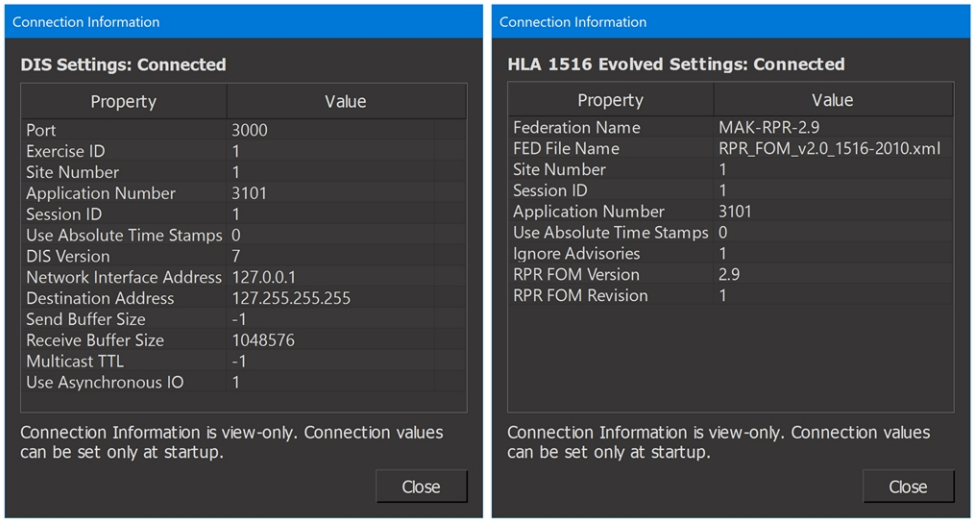
1. Choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Configure VR-Forces Connections shortcut. The Simulation Connections Configuration dialog box opens.
2. In the list of connections, click the star in the Auto Start column (★) to clear it.

4.7.7 Displaying Connection Information

You can display connection information while you are running VR-Forces. This may be useful for diagnostic purposes.

➤ To view connection information, choose **Settings > Connection Information**. The Connection Information dialog box opens (Figure 29).

Figure 29. Connection Information dialog box



4.8. Configuring Applications to Launch

By default, the VR-Forces Launcher is configured to run the VR-Forces GUI and sim engine. using the advanced options, you can specify whether those run and choose either the Simulation Object Editor (SOE) or VR-Forces Message Dump.

➤ **To select an application that you want to run:**

1. In the Simulation Connections Configuration dialog box, click the arrows to expand the advanced options (Figure 27 on page 148).
2. Select the check box next to the application name.

Using the Simulation Connections Configuration dialog box, you can add other applications to run from the Launcher and specify how they will run.

4.8.1 Adding an Application

You can edit the applications that will launch and add new ones. There must be an executable defined for the selected connection protocol. If there is no executable configured for a selected application, the Launch Selected button is disabled. For example, DIS Spy does not have an executable for HLA.

➤ **To add an application or edit one in the list:**

1. Click the Add Application button (+) or the Edit Application button (✎). The Choose Application dialog box opens.
2. Edit the options as desired. See Table 8 for information.
3. When you are finished, click OK.

Table 8. Application parameters

Option	Description
Name	The name that displays in the list of applications you can launch. It does not need to match the name of the application. You can use this in conjunction with the `(--run | -r) "application"` command-line option, where *application* is the Name.
Group	Optionally, enter a group. Applications that are in groups are collected together in the list and can be run from the command line. For details, see "4.8.1 Adding an Application" on the previous page.
Supported Command Line Options	Select the application command-line options you can use for the application, if desired. Most MAK ONE applications use the VR-Link options and you can specify those. VR-Forces applications have others you can use.
Protocol Support Strategy	Some applications have separate executables for the different protocols, while others have one executable that uses command-line options to specify the protocol. Select the option that applies to the application you selected.
Executables	If the application has separate executables for each protocol, use the Browse button (...) next to each one (DIS, HLA 1516e, and HLA 4) to navigate to the executable's location. If the application has a single executable, click the Browse button (...) and then navigate to the executable's location.
Command Line Options	Specify the command-line options to customize how the application will run. For lists of command-line arguments for VR-Forces, see "Chapter 5. Command-Line Options" on page 163.

4.8.2 Managing the Applications that Will Launch

Using the advanced options in the Simulation Connections Configuration dialog box (Figure 27 on page 148), you can specify how many instances of each application will be launched as well as the site and application IDs to use. To learn more about these IDs, see "3.2.2 How VR-Forces Simulation Engines Are Identified" on page 115.

➤ **To edit the number of instances to run, the site ID, or the application ID:**

1. Double-click on the number in the row. The field becomes editable.
2. Enter the new value. For example, you can start two instances of the sim engine by double-clicking the 1 in the Number column of the application's row and entering 2.
3. Press **Tab** or click elsewhere.

Table 9. Application launch parameters

Parameter	Command-line equivalent	Description
Number	Call the application executable more than once.	Specifies the number of instances of the
Site ID	`(--siteId | -s) ID`	Specifies the site ID. "5.4.1 Specifying the Site ID and Application Number" on page 189
Default Application ID	`(--appNumber | -a) number`	Specifies the application number. When launching multiple applications, this is the first application number to use. If you make a change, the Next Application ID immediately updates.
Next Application ID	—	Indicates the next application number that will be used. If you have used the Launch button to start several instances and you want to reset this value, click the Reset Next Application ID button (↶).
Launch	—	Click the Launch button (▶) to launch the particular application and keep the Simulation Connections Configuration dialog box open.

For each instance of the sim engine running on a machine, the application ID increments automatically to ensure there are no conflicts.

4.8.3 Using Application Groups

The list of applications shown in the advanced can be collected into groups. By default, the VR-Forces GUI and VR-Forces Simulation Engine are not in a group, while the Simulation Object Editor and VR-Forces Messages Dump are in the Other group. The group is defined in the application options.

From the command line, you can run all of the applications that are in a group. For applications with multiple protocol options, it uses the connection protocol most recently used. For example, this runs all of the applications in the Other group:

```
vrfLauncher --runGroup Other
```

4.9. Managing Plug-ins

One way to extend the functionality of VR-Forces is by loading plug-ins that have been written for it by MAK or other software vendors. You can manage which plug-ins are loaded when VR-Forces starts. You can also view information about the plug-ins that have been loaded. This is primarily used for sim engine plug-ins.

Plug-ins are implemented as one or more dynamic linked libraries (DLL or SO files). The DLLs can be release or debug, for vrfSim or vrfGui, for Windows or Linux, and for a specific protocol or protocol-independent. Each association of an application, platform, build type, and transport type is called a configuration. Use the Plug-ins Editor to specify the DLLs for use with each of the desired configurations.

- Changes to plug-in configurations do not take effect until the next time you start the affected VR-Forces application.
- You can also specify plug-ins on the command line using the `--loadPlugin` *`plugin_filename`* option. Be sure that you do not load the same plug-in multiple times.

4.9.1 Loading Plug-ins

You can specify which plug-ins to load in either of the following ways:.

- In the Plug-ins dialog box, specify the plug-ins to load the next time VR-Forces starts.
- In the Simulation Connections Configuration dialog box, specify the plug-ins to load at startup.

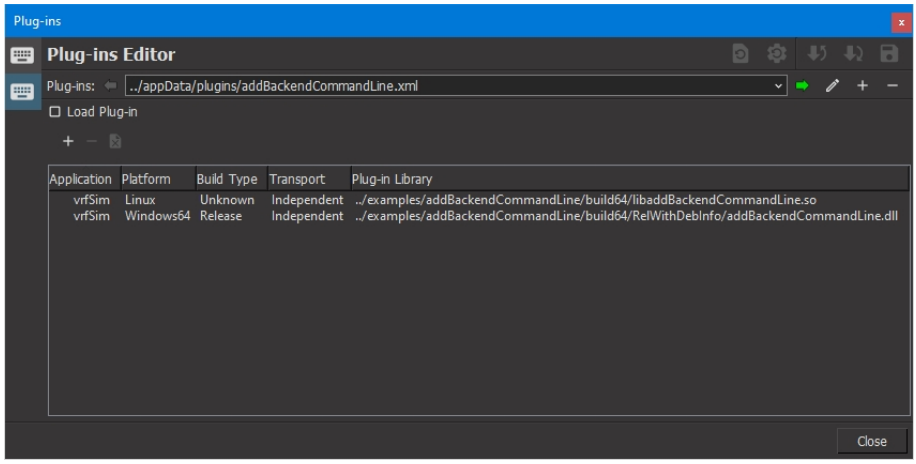
Specifying the Plug-ins to Load in the Plug-ins Dialog Box

Use the Plug-ins dialog box to specify the plug-ins that will load the next time that you start VR-Forces.

➤ **To specify that a plug-in be loaded or not loaded in the Plug-ins dialog box:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 30).

Figure 30. Plug-ins dialog box, Plug-ins Editor page



3. In the list of plug-ins at the top of the page, select the one that you want to edit. The configurations for that plug-in are listed in the window.
4. Select or clear the Load Plug-in check box.

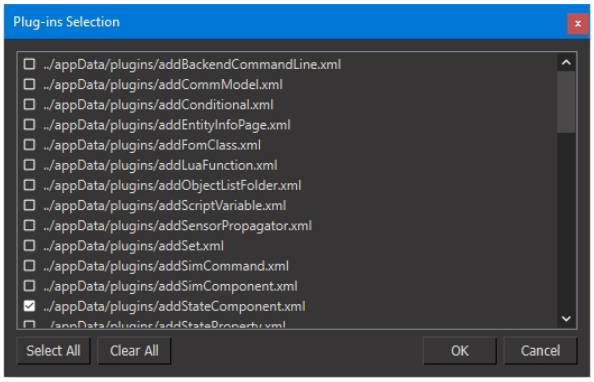
Specifying the Plug-ins to Load in the Simulation Connections Configuration

You can select the plug-ins that you want to load at startup in the launcher.

➤ **To specify the plug-ins to load in the simulation connections configuration:**

1. Start VR-Forces using the VR-Forces Launcher, as described in "4.1.1 Starting VR-Forces from the VR-Forces Launcher" on page 129. The Simulation Connections Configuration dialog box opens (Figure 27 on page 148).
2. Click Plug-ins. The Plug-ins Selection dialog box opens (Figure 31). The dialog box lists all of the plug-ins that are configured for VR-Forces. If a plug-in has previously been selected for loading, in this dialog box or in the Plug-ins Editor, it is checked.

Figure 31. Plug-ins Selection dialog box



3. Select the check boxes for the plug-ins that you want to load. Clear the check boxes for any plug-ins that you do not want to load.
4. Click OK. The plug-in configuration applies to all of the startup configurations.

4.9.2 Adding a Plug-in

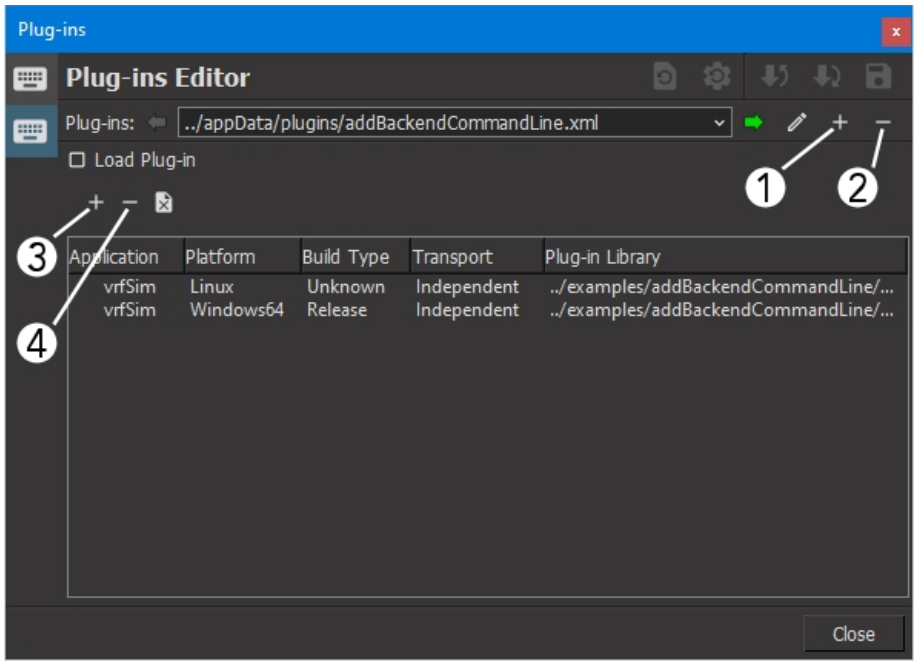
When you add a plug-in, VR-Forces creates an XML file (in *./appData/plugins*) that contains all of the information VR-Forces needs to load the plug-in. New plug-ins are added with entries for each of the configurations (application, platform, build type, transport). After you add a plug-in, you must specify the DLLs for each configuration that you want to support.

➤ **To add a plug-in:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 30 on the previous page).
3. Click the Add button (+) at the upper right corner of the page (#1 in Figure 32). The New Plug-in dialog box opens.
4. Type a name for the plug-in.

5. Click OK. The plug-in is added to the Plug-ins list. It has configurations for each of the combinations of application, platform, build type, and transport (Figure 32) as shown in the figure.

Figure 32. New plug-in



6. Specify the DLL for each configuration that you want to support, as described in "4.9.3 Specifying the DLLs for a Plug-in" below.

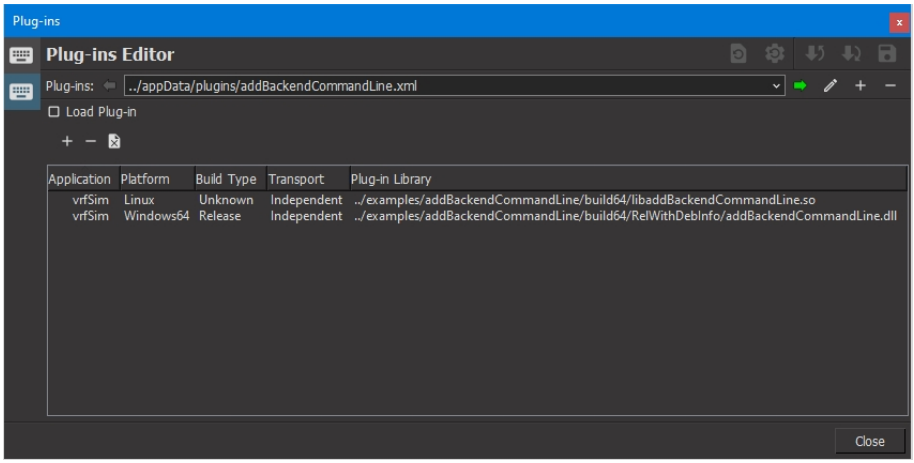
4.9.3 Specifying the DLLs for a Plug-in

Each plug-in configuration that you want to be able to load must specify the DLL to load for that configuration.

➤ **To specify the DLL for a plug-in configuration:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 33).

Figure 33. Plug-ins Editor



3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Click the Plug-in Library column for the configuration that you want to edit. The Select Plug-in dialog box opens.
5. Select the DLL that you want to load for this configuration.
6. Click Open. The value changes.

Removing a Plug-in's DLL

You can remove the DLL assignment for a plug-in configuration without removing the configuration.

➤ **To remove the DLL assignment for a plug-in:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 33).
3. Select the configuration whose DLL assignment you want to remove.
4. Click the Erase button ( ).

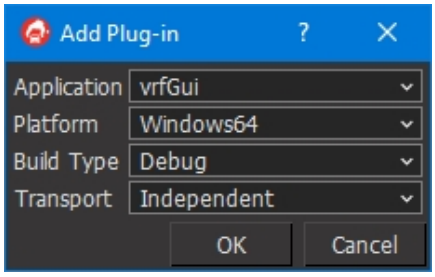
4.9.4 Adding a Plug-in Configuration

A plug-in configuration is a row in the Plug-ins Editor window. It specifies an application, platform, build type, transport type, and plug-in library.

➤ **To add a plug-in configuration:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 30 on page 157).
3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Click the Add button ( ) above the list of configurations (#3 in Figure 32 on the previous page). The Add Plug-in dialog box opens (Figure 34).

Figure 34. Add Plug-in dialog box



5. For each parameter (Application, Platform, Build Type, and Transport), select a value from the list.
6. Click OK. The configuration is added to the list.
7. Specify a DLL for the configuration, as described in "4.9.3 Specifying the DLLs for a Plug-in" on page 159.

4.9.5 Deleting a Plug-in Configuration

➤ **To delete a plug-in configuration:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 30 on page 157).
3. In the list of plug-ins at the top of the page, select the one that you want to edit.
4. Select the configuration that you want to delete.
5. Click the Delete button (—) that is above the list of configurations (#4 in Figure 32 on page 159).

4.9.6 Deleting a Plug-in

When you delete a plug-in, you delete the XML configuration file in *./appData/plugins* that specifies the plug-in configurations and whether to load it.

➤ **To delete a plug-in:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Editor page (Figure 30 on page 157).
3. In the list of plug-ins at the top of the page, select the one that you want to delete.
4. Click the Delete button (—) to the right of the plug-ins list (#2 in Figure 32 on page 159).

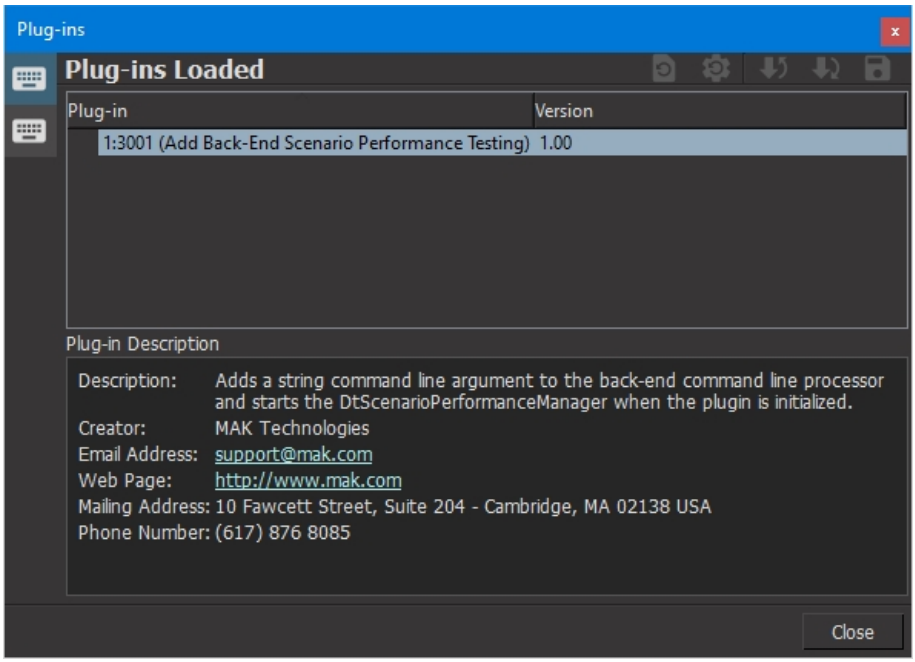
4.9.7 Viewing a List of Loaded Plug-ins

You can view a list of the plug-ins that will load when you start VR-Forces. Plug-ins may include a description that you can also view.

➤ **To view the details of loaded plug-ins:**

1. Choose **Settings > Plug-ins**. The Plug-ins dialog box opens.
2. Select the Plug-ins Loaded page (Figure 35).

Figure 35. Plug-ins dialog box, Plug-ins Loaded page



3. To view details about a plug-in, select it in the list.

4.10. The VR-Forces Log Files

On Windows, VR-Forces creates two log files, *vrfSim.log* and *vrfGui.log*. Both log files are written to the *logs* folder. The default location is *C:/MAK/logs*.

On Linux, in combined mode, VR-Forces automatically creates a log file, *vrfSim.log* for sim engine output. GUI output is sent to the console. If you want to view sim engine output, open a second console window, and do `tail -f vrfSim.log`. If you want to send GUI output to log files, or if you are running in separate mode and want to create a log file for either the GUI or sim engine, you need to redirect your output to a file, for example:

```
vrfGui > vrfGui.log
```



5. Command-Line Options

VR-Forces has command-line options for vrfGui and vrfSim.

5.1. Command-Line Options for vrfGui 164
5.2. Command-Line Options for vrfSim 178
5.3. Command-Line Options for vrfLauncher 187
5.3.1 Running in Combined Mode from the Command Line 188
5.3.2 Using the additionalargs-- Command-line Argument 188
5.4. Protocol-Independent Command-line Options 189
5.4.1 Specifying the Site ID and Application Number 189
5.4.2 Specifying the Language to Use in the GUI 189
5.4.3 Setting the Notification Level 189
5.4.4 Loading Plug-ins 190
5.5. Command-Line Options for HLA Federations 190
5.5.1 Specifying the Federation Execution 190
5.5.2 Specifying the FED File 191
5.5.3 Specifying the RPR FOM Version 191
5.5.4 Specifying a FOM Mapper 191
5.5.5 Specifying the RPR FOM Revision 191
5.5.6 Specifying FOM Mapper Initialization Data 191
5.5.7 Specifying a FED File That Is Appropriate for the FOM Mapper 192
5.5.8 Enabling Time Management 192
5.6. Command-Line Options for DIS Exercises 192
5.6.1 Specifying the Port Number 192
5.6.2 Specifying Point-to-Point or Multicast Operation 192
5.6.3 Specifying the Multicast Time-to-Live 193
5.6.4 Using Asynchronous I/O 193
5.6.5 Specifying the Exercise ID 193
5.6.6 Specifying the DIS Version 193

5.1. Command-Line Options for vrfGui

Table 10 lists VR-Forces's command line options for vrfGui (the VR-Forces GUI).

Table 10. vrfGui command-line options

Option	Description
`--antiAliasing` *level*	Set the anti-aliasing level (0, 2, 4, 8, or 16). Default: 4. Depending on your graphics card and driver, some combinations of anti-aliasing and bit settings (`--depthBits`, `--stencilBits`, and `--alphaBits`) may be incompatible. For example, if you set stencil bits to 8, the depth must be set to 24 and anti-aliasing may need to be set to 0 or 2.
`--appCacheDir` *directory*	Use `--appCacheDir` to specify a directory for the cache. This allows you to run other instances of VR-Forces from the same installation. Running two RDEs on the same machine helps compensate for the limited threading in VR-Forces.
`--appDataDir` *directory*	Specifies the location of application data. If not specified, VR-Forces uses the default: *./appData*.
(`--autoConnect` | `-C`) {*filepath* | *connector_name*}	Automatically connect to the network using the specified connector. Can be used multiple times.
`--dataDir` *directory*	Specifies the location of the application's data directory. If not specified, VR-Forces uses the default, *./data*.
`--defaultSimulationModelSet` *sms*	The simulation model set to use as the default when starting up the GUI.
`--depthBits` *bits*	Sets the depth of graphics to 24 bits or 32 bits. The default is 24, This has no effect when reverse Z is enabled, which is the default for VR-Forces. The number of depth bits in reverse Z is always 32 bits float.
`--dis`	Start in DIS mode, using DIS command-line arguments to specify connection parameters. DIS-specific arguments do not have to follow `--dis`; they can be in any order.

Table 10. vrfGui command-line options (continued)

Option	Description
`--disableDynamicTerrain`	Disables the dynamic terrain engine. May slightly speed up terrain loading and paging.
`--disableErrorDialog`	Prevents error messages from displaying. Errors print to the console instead.
`--disableGlBufferTracking`	Disable GL buffer tracking, which can improve performance in cases such as CDB terrains.
`--disableIndirectEntities`	Disables indirect rendering of entities. This is primarily for debugging and fixing any regression with the old path. It may or may not match the behavior with the default path (which is indirect).
`--disableIndirectExternalRenderer`	Disables the indirect external renderer, which minimizes state changes by default. When this is used, state changes are not minimized.
`(--disableKDTrees | -K)`	Disable creation of KD trees for intersections.
`--disableModelSets "`*`model-set-number`*`"`	Disables one or more model sets. Model set numbers are: • `0` - 3D Model Set • `1` - Colorized Models Model Set • `4` - Minimap • `5` - 2D Icons Model Set To disable multiple model sets, separate them with commas. For example: `--disableModelSets "0, 1"` disables the 3D and XR model sets `--disableModelSets "0, 1, 5"` disables all model sets
`(--disableReverseZBuffer | -z)`	Disables use of reverse Z buffering, which is enabled by default. Leaving reverse Z buffering enabled can reduce Z-fighting when viewing the terrain from a distance. (3D observer modes only.)
`--disableTerrainObserverViews`	Disables the loading of terrain observer view files. Loading a terrain does not clear the observer views and the terrain's observer views do not load.
`--dispSetting` *`filename`*	Specifies the display engine configuration to load.

Table 10. vrfGui command-line options (continued)

Option	Description
`--doNotCheckPluginVersions`	Do not check the version in a plug-in to make sure it can load in VR-Forces. Developers build plug-ins against a particular version of VR-Forces. Ordinarily, when VR-Forces loads a plug-in, it checks to be ensure whether it was created with that version of VR-Forces. This option prevents that check.
`--doNotLoadVrfPlugins`	Prevents loading of plug-ins in the *./plugins* directory.
`--enableDofConstraints`	Enables the use of degree(s) of freedom constraints on simulation object models.
`--enableLogFileTimestamps`	Print a timestamp for each line in a log file.
`--enablePersistentShadowTechniq ue`	Use this command-line option to enable persistent shadows. By default, shadows are disabled: the shadow camera traversal is skipped but the shaders remain the same, preventing additional shader sets in SpeedTree and instant switching. If the VP for the shadow pass is always installed, SpeedTree (and other drawables) do not regenerate VP shaders/user shaders.
`--enableTracyProfiling`	Enables the Tracy Client when you launch the application.
`--enableVSync`	Turns on VSync.
`--entDispSetting` *filename*	Specifies the entity display setting to load.
`--entityDefsDir` *directory*	Loads the *.hier* and *.leaf* files in the specified directory to the entity element definition manager.

Table 10. vrfGui command-line options (continued)

Option	Description
(--entityGridCellSize *x y z* | --entityGridCellSize *x,y,z* | --entityGridCellSize *integer*)	Use this option for debugging, particularly looking for bottlenecks in the code. Entities (in indirect) are divided into a grid of cells. The cells have the same size. This is a regular (sparse) grid that allows for efficient, higher-level culling. The default cell size is 5000 5000 5000. You can specify three values or one, for example: • 2000 2000 2000 • 2000 • 2000 200 10000 • -1 -1 -1 (one cell, infinite cell size) • -1 (one cell, infinite cell size)
(--entTypeMap | -E) *filename*	Load the specified object type mapping file. The filename extension determines the type of mapping.
--envSetting *filename*	Specifies the environment configuration file to load.
--exConnConfigFile *filename*	Specifies the exercise connection configuration file to use. Use this to override settings in the VR-Link exercise connection configuration file, *./appData/settings/connections/MAK-ONE-YYYY-Config.xml*, which defines the default connections settings. For more information, refer to the *MAK ONE Interoperability Guide*.
(--extraArgs | -e) *string* | "*string1 ... stringn*"	Allows you to send arbitrary strings to the initializer, which can be queried for by plug-ins. This option works around the restriction on adding new command-line arguments. It can take one argument or multiple arguments. Multiple arguments must be wrapped in double quotes.
--factoryRootDir *directory*	Set the root directory used to restore settings to factory defaults.
--forceTextShaping	Force all text to use HarfBuzz shaping. Use this option if you need support for right-to-left languages. It may affect performance.
--fowPerspective *force*	Shows the ground truth and spot reports perspective of the specified force, where *force* can be Friendly, Opposing, Neutral, or None. Default: None.

Table 10. vrfGui command-line options (continued)

Option	Description
`--fullScreen`	Start with the window in full screen mode.
`--gui` *filename*	Specify a GUI configuration file. This file specifies the layout of menus, toolbars, and dialog box pages. A simple filename assumes *./appdata/settings/vrfGui*. A relative filename is from the *./bin*64 directory. A full path is used as is.
(`--hasNTPSync` | `-O`)	Enables the synchronization of all time-based features between different VR-Vantage applications, such as instances of VR-Vantage or the VR-Forces GUI. Note that system clocks must be synchronized. Time-based features include the rendering of light point animations, continuous flipbook animations, ocean waves, clouds, and SpeedTrees.
`--highestPriority`	Changes the priority of the application and the main render thread to the highest available priority. This command may starve other processes.
`--hla1516e`	Start in HLA Evolved mode, using HLA command-line arguments to specify connection parameters. HLA-specific arguments do not have to follow `--hla1516e`; they can be in any order.
`--hla4`	Start in HLA 4 mode, using HLA command-line arguments to specify connection parameters. HLA-specific arguments do not have to follow `--hla4`; they can be in any order.
(`--ignoreZoomForSpeedtreeLod` | `-Z`)	Disable zoom for SpeedTree LOD.
`--layoutSettingsFile` *filename*	Specifies the GUI layout settings file to use. *filename* specifies the base filename. It is prefixed with *default_* and the extension *.uisx* is added. Default: UILayoutSettings.
ⓘ Use the following texture limiting command-line options to limit and debug the memory usage and then observe the total GPU memory used.	
`--limitEarthSplattingTextures` "*width and height*"	Restricts the size of an earth file's splatting textures to the specified resolution. The width and height must be a power of 2, for example, "16". If you do not enter a power of 2, VR-Forces converts it to the next highest power of 2.

Table 10. vrfGui command-line options (continued)

Option	Description
`--limitEarthTerrainTextures "*width and height*"`	Restricts the size of an earth file's terrain textures to the specified resolution. The width and height must be a power of 2, for example, "16". If you do not enter a power of 2, VR-Forces converts it to the next highest power of 2.
`--limitEarthVegetationTextures "*width and height*"`	Restricts the size of an earth file's vegetation textures to the specified resolution. The width and height must be a power of 2, for example, "16". If you do not enter a power of 2, VR-Forces converts it to the next highest power of 2.
`--limitEntityTextures "*width height*"`	Uses only mipmap levels at or below the resolution for entities. Use two integers for the maximum width and height of the texture mipmap to use, for example, "1024 1024" or "2k 2k".To turn off the texture limit function, use `--limitEntityTextures "-1 -1"`.
`--limitEntityTexturesExclusions "*texture*"`	Excludes textures from texture reduction limits for indirect. Specify the texture(s) that you want to exclude, for example, `--limitEntityTexturesExclusions "SurfaceCVN-75HarrySTruman, SurfaceCruiseShip, FixedWingF-16, FixedWingRQ-1Predator"`.
`--limitTerrainTextures "*width height*"`	Uses only mipmap levels at or below the resolution for terrain. Use two integers for the width and height, for example, "1024 1024" or "2k 2k".

In most situations, using `--limitEntityTextures` and `--limitTerrainTextures` prevents crashes with high VRAM. If you continue to experience crashes, use `--limitGpuMemoryPercentage` in addition to these command-line options.

Table 10. vrfGui command-line options (continued)

Option	Description
`--limitGpuMemoryPercentage` *integer*	Set the cutoff percentage as a value between 0 and 100. (Typical values are between 85 and 100.) Beyond the specified percentage, VR-Forces does not make new textures or buffers as a percentage of the available video RAM. This applies only to VR-Forces-side textures: terrain insets, entities, indirected osgEarth layers (feature_geom and buildings). It does not apply to SilverLining, Triton, or osgEarth tiles. The latter are controlled by the *.earth* file options.
Typically, you would use `--limitGpuMemoryPercentage` in conjunction with `--limitEntityTextures` and `--limitTerrainTextures` when you see crashes with high VRAM.	
`--loadPlugin` *plugin_filename*	Specifies a plug-in to load. Accepted multiple times. The filename can be a DLL or SO or an XML file that specifies the plug-ins to load. The order in which you list multiple plug-ins on the command line does not determine the order in which they are loaded. For details, see "4.9. Managing Plug-ins" on page 156.
`--loadSimulationObjectGroup` *group*	Creates a new scenario on the simulation object group editing terrain and creates an instance of the simulation object group on that terrain.
(`--locale` | `-G`) *language*	Specifies the locale (for localization).
(`--logFileName` | `-l`) *filename*	Specifies the log file name.
`--mainScreenNum` *num*	Specifies the display that the main window should use. The screen number can be a number from 0 to the number of screens -1. This number matches the screen number used in the Display Engine Configuration Editor for a window.
`--mergeEntityTypeMap` {*directory* | *filename*}	Merges all entity type mapping files (**.metx*) in a directory or individual entity type mapping files. Can be used multiple times. See note below.

Table 10. vrfGui command-line options (continued)

Option	Description
`--mergeHierarchyEntityDef {`*`directory`* `|` *`filename`*`}`	Merges the *.hier* files in the specified directory or the individual *.hier* file into the default hierarchy (usually loaded from *./SharedData/19/latest/ModelData /Configuration/Original/Definitions*). Merging is additive only. No nodes in a previously loaded hierarchy are replaced. Can be used multiple times.
`--mergeHierarchySpotReportDef {`*`directory`* `|` *`filename`*`}`	Merges the element definition *.hier* files in the specified directory or the individual *.hier* file into the hierarchy after *./SharedData/19 /latest/ModelData/Configuration/Original/Definitions* has been loaded. Can be used multiple times.
`--mergeHierarchyTacticalGraphicsDef {`*`directory`* `|` *`filename`*`}`	Merges the specified *.hier* file(s) into the default hierarchy (usually loaded from *./SharedData/19/latest/ModelData/ Configuration/Original/Definitions*). Merging is additive only. No nodes in a previously loaded hierarchy are replaced. Can be used multiple times.
`--mergeHierarchyUnitDef {`*`directory`* `|` *`filename`*`}`	Merges the specified *.hier* file(s) into the default hierarchy (usually loaded from *./SharedData/19/latest/ModelData/ Configuration/Original/Definitions*). Merging is additive only. No nodes in a previously loaded hierarchy are replaced. Can be used multiple times.
`--mergeLeafEntityDef {`*`directory`* `|` *`filename`*`}`	Merges the leaf files in the specified directory or the individual leaf file into the default entity element definitions. Can be used multiple times.
`--mergeLeafSpotReportDef {`*`directory`* `|` *`filename`*`}`	Merges the leaf files in the specified directory or the individual leaf file into the default spot report element definitions. This adds or replaces leaves. Can be used multiple times.
`--mergeLeafTacticalGraphicsDef {`*`directory`* `|` *`filename`*`}`	Merges the leaf files in the specified directory or the individual leaf file into the default tactical graphics element definitions. Can be used multiple times.

Table 10. vrfGui command-line options (continued)

Option	Description
`--mergeLeafUnitDef {`*`directory`* `|` *`filename`*`}`	Merges the leaf files in the directory or the specified individual leaf file into the default unit element definitions. Can be used multiple times.
`--mergeModelDefs {`*`directory`* `|` *`filename`*`}`	Merges all model definition files (**.ommx*) in the specified directory or individual model definition files. Can be used multiple times.
`--mergeTacticalGraphicsTypeMap {`*`directory`* `|` *`filename`*`}`	Merges all tactical graphics type mapping files (**.mtgx*) in a directory or individual tactical graphics type mapping files. Can be used multiple times. See note below.
`--mergeUnitTypeMap {`*`directory`* `|` *`filename`*`}`	Merges all unit type mapping files (**.magx*) in a directory or individual unit type mapping files. Can be used multiple times. See note below.

The options for merging type maps (--mergeEntityTypeMap, --mergeTacticalGraphicsTypeMap, and --mergeUnitTypeMap) allow you to merge your type mapping with those that already exist. By default, that does not overwrite the existing mappings. Your merged mappings will exist alongside those that were already loaded.

If you want to overwrite the existing ones, use the merge option alongside the --overwriteMappingsOnMerge command-line option.

`--noAppDataEntityDefs`	Do not load the entity element definitions in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataEntityTypeMap`	Do not load entity type mappings in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataModelDefs`	Do not load model definitions in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataSpotReportDefs`	Do not load the spot report definitions in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataTacticalGraphicsDefs`	Do not load the tactical graphics element definitions in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataTacticalGraphicsTypeMap`	Do not load tactical graphics type mappings in *./SharedData/19/latest/ModelData/Configuration/Definitions*.

Table 10. vrfGui command-line options (continued)

Option	Description
`--noAppDataUnitDefs`	Do not load the unit element definitions in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--noAppDataUnitTypeMap`	Do not load unit type mappings in *./SharedData/19/latest/ModelData/Configuration/Definitions*.
`--nonVrfObjectsUUIDScheme` *scheme*	Sets the UUID scheme for non VR-Forces objects. Values are `entity-identifier`, `global-identifier`, `marking-text`. Default: entity-identifier.
(`--notifyLevel` | `-n`) *notification_level*	Specifies the notification level for application messages, where *notification_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, see "5.4.3 Setting the Notification Level" on page 189.
`--overwriteMappingsOnMerge`	Any mapping merge specified on the command line overwrites mappings in the default files. Use in conjunction with the command-line options for type maps (above).
`--plugin` *filename*	Load the specified GUI plug-in. Can be used multiple times.
`--pseudoFullScreen`	Use pseudo full screen for full screen mode. This may improve performance when using dynamic ocean.
`--scenarioFile` *filename*	Specifies a scenario to load.
`--sharedDataDir` *directory_path*	Use this to specify the location of the shared data directory if you installed it somewhere other than the default, which is *./SharedData*.
`--sharedDataRoot` *directory_root*	Specifies the location of the shared data root. This combined with the stem and data version creates the shared data path.
`--sharedDataStem` *directory_stem*	Specifies the location of the shared data stem. This combined with the data root and data version creates the shared data path.
`--sharedSettingsDirectory` *directory*	Specifies the directory where shared settings are stored. The default directory is *$(APP_DIR)/settings*. If the `--useUserSettingsDirectory` flag is set, then the user settings directory is used to store the settings instead.
(`--showConsole` | `-c`)	Display the console window.

Table 10. vrfGui command-line options (continued)

Option	Description
`--stencilBits` *bits*	Sets the number of stencil bits (0 or 8). Default: 0.
(`--suppressHatIntersectOnAttach` | `-S`)	Suppress the periodic HAT intersection when attached. This setting can improve performance if you need to attach the observer to ground vehicles a lot, but do not need to attach to air vehicles.
`--tacticalGraphicsDefsDir` *filename*	Loads the *.hier* and *.leaf* files in the specified directory to the tactical graphics element definition manager.
terrain_MTF_file	Specifies a terrain (MTF) file to load. By default, VR-Forces looks in the expected data location for terrain files in either the default location, *C:\MAK\SharedData*, or the one specified by the environment variables MAK_SHARED_DATA_ROOT and MAK_SHARED_DATA_STEM. (If the file is in another location, the path must be a relative path from *./bin64* to the file, for example, *../../myTerrains/MTF_file*.)
`--unitDefsDir` *filename*	Loads the *.hier* and *.leaf* files in the specified directory to the unit element definition manager.
`--useFileCommChannel` *channel*	The communication file to be used as a base for communicating text-based commands from external applications.
`--useUserSettingsDirectory`	Indicates that the system should store settings in the user login settings directory, rather than the application settings directory. Use in conjunction with `--sharedSettingsDirectory`.
`--userDataDir` *directory*	Specifies the location of user data directory. If not specified, VR-Forces uses the default: *./userData*.
`--userScenarioDirectory` *directory*	Specifies the user scenario directory to consider the "home", for example, `../userData/scenarios/Sample`. If not specified, the home directory is *./userData/scenarios*.
`--userTerrainDirectory` *directory*	Specifies the user terrain directory. If you do not define a different directory, the application uses the default: $(SHARED_DATA_DIR)/Terrain/TerrainConfiguration.

Table 10. vrfGui command-line options (continued)

Option	Description
(--ignore_rest | --)	Ignores all command-line options that are listed after this argument. This option is useful for disabling part of a command line in a script for test purposes so that you do not have to retype all of the command options at a later point.
(--version | -v)	Displays version information.
(--help | -h)	(Displays command-line usage information in a console window.)
HLA Only	**These options are available when you specify --hla1516e or --hla4.**
(--appNumber | -a) *number*	Specifies the application number.
--disableRequestUpdates	Disable request updates for state objects.
--disableRemoteDiscovery	Disable discovery of remote objects.
(--execName | -x) *exec_name*	Specifies the name of the federation execution. Default: MAK-RPR-2.0.
(--federateName | -N) *federate_name*	Specifies a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI automatically assigns a unique name for the federate. Default: VR-Forces. (HLA Evolved only)
(--federateType | -p) *type*	Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.
(--fedFileName | -F) *fedfile_name*	FED file name.
--fileTransporterReceivePort *port*	Port for the file transporter. This is the port used to receive order of battle and plan files.
--fomMapperInitData *data*	FOM Mapper initialization data.
(--fomMapperLib | -f) *libname*	FOM Mapper library name.
--fomModules *module*	Specifies an HLA Evolved or HLA 4 FOM module. Can be used multiple times.
(--hostAddressString | -H) *address*	Specifies the host address. This is usually the same as the device address.
--ignoreAdvisories	Enables or disables HLA advisory messages. Default: true.

Table 10. vrfGui command-line options (continued)

Option	Description
(--localSettingsDesignator | -S) *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
--numberOfremoteNetworkThreads *thread_count*	For remote object discovery, the number of worker threads to use to help discover remote objects for data usage.
--processStateDataUponDiscovery	Processes state data when entities are discovered rather than waiting for a specific length of time between discovery.
--reportedHostAddressString *address*	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers. It is only important when you have a GUI or remote controller on one side of a router and a simulation engine on the other, and there is some kind of network address translation occurring whereby they have to connect to one another using an IP address other than the one they recognize themselves at locally.
--requestWhenInitialized	Request attribute updates only when a state object is initialized.
--rprFomRevision *revision_number*	RPR FOM revision for RPR FOM 2.0, draft 17.
--rprFomVersion *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
(--sessionId | -i) *ID*	Specifies the session ID for this instance of the GUI.
(--siteId | -s) *ID*	Specifies the site ID.
--useAbsoluteTimeStamps	Enables use of absolute timestamps. Otherwise uses relative timestamps.
DIS Only	**These options are available when you specify --dis.**
(--appNumber | -a) *number*	Specifies the application number.
--appIdRange *range*	The application ID range for VR-Forces sim engines.
--defaultObjectTimeout *timeout*	Specifies the timeout interval for objects.
(--destAddrString | -A) *address*	Specifies the destination address.

Table 10. vrfGui command-line options (continued)

Option	Description
`--deviceAddress` *address*	Specifies the address of the network card to use for UDP traffic.
`--disableRemoteDiscovery`	Disables discover of remote simulation objects.
(`--disPort` | `-P`) *port*	DIS port.
`--disVersion` *version*	The DIS version. Default: 7.
(`--exerciseId` | `-x`) *ID*	Exercise ID.
`--fileTransporterReceivePort` *port*	Port for the file transporter. This is the port used to receive order of battle and plan files.
(`--hostAddressString` | `-H`) *address*	Specifies the host address. This is usually the same as the device address.
`--mcastTtl` *time*	Specifies the multicast time-to-live. Default: -1.
(`--multicastAddresses` | `-S`) *address1 ... addressn*	Specifies one or more multicast addresses.
`--numberOfRemoteNetworkThreads` *threads*	When discovering remote objects, specifies the number of worker threads to use to help discover remote objects for data usage.
`--recvBufferSize` *size*	Receive buffer size.
`--reportedHostAddressString` *address*	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers. It is only important when you have a GUI or remote controller on one side of a router and a simulation engine on the other, and there is some kind of network address translation occurring whereby they have to connect to one another using an IP address other than the one they recognize themselves at locally.
`--sendBufferSize` *size*	Send buffer size. Default: -1.
(`--sessionId` | `-i`) *ID*	Specifies the session ID for this instance of the GUI.
(`--siteId` | `-s`) *ID*	Specifies the site ID.
`--useAbsoluteTimeStamps`	Enables use of absolute timestamps. Otherwise uses relative timestamps.
`--subnetMask` *mask*	Specifies a subnet mask.
`--useAsyncIO`	Use asynchronous I/O. Default: False.
`--useIpv6`	Specifies that the DIS exercise connection use the IPV6 protocol, if available.

5.2. Command-Line Options for vrfSim

Table 11 describes the command-line options for vrfSim (the VR-Forces sim engine).

Table 11. vrfSim command-line options

Option	Description
`--additionalSystemScriptsDirectory` *directory*	Specifies a directory where VR-Forces will look for system scripts when it populates the Scripted Task dialog box script list.
`--appDataDir` *directory*	Specifies the location of application data. If not specified, VR-Forces uses the default: *./appData*.
`--appIdRangeMax` *max*	Specifies a range maximum to roll the application number to the next ID in the range. When the upper limit of the entity number is reached, the application number for the next object created will be increased and the entity number restarted at 0. If the application number range max is reached no more objects will be created on the sim engine until some are deleted.
(`--appNumber` | `-a`) *number*	Specifies the application number. For details, see "5.4.1 Specifying the Site ID and Application Number" on page 189.

Table 11. vrfSim command-line options (continued)

Option	Description
`--attributePduStrategy`	This flag controls how VR-Forces sends DIS attribute PDUs. For the GUI to control VR-Forces Entities, the Simulation Engine must supply additional attributes through DIS. Some applications cannot correctly interpret coupled attribute PDUs in some DIS versions. In those cases, changing the default behavior may be required.The possible values are: • 0: DtAttributePduStrategy_ Unspecified • 1: DtAttributePduStrategy_ NoAttributePdu • 2: DtAttributePduStrategy_Couple • 3: DtAttributePduStrategy_ DoNotCouple • 4: DtAttributePduStrategy_ DoNotCoupleAndSendAttributeFirst The default is 0, which means that VR-Forces chooses based on the selected version of DIS. For DIS 7, it uses 2. Mode 1 will not send any additional attributes, so VRF GUI will not recognize the published entities as VRF entities and will not be able to control them.
`(--batchScenarioFileName | -B)` *batch_file*	Runs the specified batch file (for use with one sim engine only). For more information, see "Running in Batch Mode from the Command Line" on page 272.
`--clockMode` *mode*	For Windows only, specifies whether to use the performance counter or the system clock for absolute time stamping. This only applies when Use Absolute Time Stamps is enabled. **0** - Performance counter (default), highly accurate internal timer **1** - Continual synchronization with system clock
`--connectionless`	Run VR-Forces without a connection.
`--countryCodesMappingFile` *file_name*	Specifies an alternative file that maps country codes to country names and DIS enumeration values.

Table 11. vrfSim command-line options (continued)

Option	Description
`--daemon`	(Linux only) Starts the sim engine as a daemon process . If you start *vrfSim* as a daemon, it does not try to read characters from the console. It forks a background process at startup. It will exit cleanly if it receives a SITGTERM, SIGQUIT, or SIGINT signal.
`--dataDir` *directory*	Specifies the location of the application's data directory. If not specified, VR-Forces uses the default, *./data*.
(`--destAddrString` | `-A`) *address*	Specifies the destination address.
`--deviceAddress` *address*	Specifies the address of the network card to use for UDP traffic.
`--dis5Timestamps`	In DIS 7, there is a discrepancy in timestamps that can lead to some of them appearing to take place in the future. Use this option to enable the time management method from DIS 5.
`--diGuyAnimationsFile` *file_name*	Specifies an additional configuration file for DI-Guy animations.
`--diGuyCharacterDataFile` *file_name*	Specifies an additional configuration file for DI-Guy character data.
`--disableParallelPublisherTick`	Turn off ticking of publishers in parallel.
`--disableParallelTick`	Execute all tick code serially.
`--disableRemoteDiscovery`	Disable discovery of remote objects.
`--doNotLoadVrfPlugins`	Prevents loading of plug-ins in the *./plugins* directory.
(`--doNotUseConsole` | `-q`)	Specifies that all *vrfSim* output go to the log file rather than the console (quiet mode). If you are using a high level of notification, sending output to the console can degrade performance. Running in quiet mode prevents this degradation of performance. This command only applies to sim engines running on Windows. To create a log file on Linux, redirect the output of vrfSim, for example: `vrfSim > mylog.txt` For more details about log files, see "4.10. The VR-Forces Log Files" on page 162.

Table 11. vrfSim command-line options (continued)

Option	Description
`--enableChannel` *channel*	Enables a specific channel for the channel-specific output, even if it is disabled in *channelSettings.mtl*.
`--enablePduProfiling` *string*	Enables the entity PDU flood check of a given time or a number of packets over that time.
`--exConnConfigFile` *filename*	Specifies the exercise connection configuration file to use. Use this to override settings in the VR-Link exercise connection configuration file, *./appData/settings/connections/MAK-ONE-YYYY-Config.xml*, which defines the default connections settings. For more information, refer to the *MAK ONE Interoperability Guide*.
`--exitOnTestResultCommand`	If a plan in the simulation includes the Test Result statement, and the simulation engine was started with this command, then the simulation sends the message to all sim engines to close after the Test Result statement. For details, see "43.7.9 Sending Test Results" on page 979.
`--fileNotifyLevel` *level*	Notify level in to use in the log file.
`--frontEndPID` *PID*	The process ID for the GUI when running in combined mode.
`--fullyLoadTerrain`	For paged terrains, load all pages.
`--hostAddressString` *address*	Specifies the host address. This is usually the same as the device address. Called Device Address in the Launcher.
`--loadPlugin` *plugin_filename*	Specifies a plug-in to load. Accepted multiple times. The filename can be a DLL or SO or an XML file that specifies the plug-ins to load. The order in which you list multiple plug-ins on the command line does not determine the order in which they are loaded. For details, see "4.9. Managing Plug-ins" on page 156.
`--logFileName` *filename*	Specifies the log file name.
`--logFrameRateStatistics`	Starts the Tracy Client connection to capture performance statistics for the sim engine. See "6.4.2 Using the Tracy Client" on page 203 to learn more.

Table 11. vrfSim command-line options (continued)

Option	Description
`--mcastTtl` *time*	Specifies the multicast time-to-live. Default: -1.
`--msdlSIDCMappingFile` *file_name*	Specifies an alternative file for mapping SIDC codes to DIS enumerations.
(`--notifyLevel` | `-n`) *notification_level*	Specifies the notification level for messages written to the console or the VR-Forces log file (*vrf.log*), where *notification_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, see "5.4.3 Setting the Notification Level" on page 189.
`--nonVrfObjectsUUIDScheme` *scheme*	Sets the UUID scheme for non VR-Forces objects. Values are `entity-identifier`, `global-identifier`, `marking-text`.
`--numCallbackThreads` *threads*	Number of threads in the callback queue.
`--numNetworkCallbackThreads` *threads*	Number of network threads in the callback queue. Zero means run network in a thread serially (unless it is turned off). If publishing is in the tick thread, this will be shared with publishers.
`--outputInterval` *interval*	Interval for output.
`--outputLogFile` *file_name*	The file to use to write out application output.
`--performanceOutput` *string*	This specifies where you want to save some output to disk. It's for internal use in nightly autotests and performance tests. I've got it here for when we check the command-line options lists.
`--recvBufferSize` *size*	Receive buffer size.
(`--scenarioFileName` | `-L`) "*scenario_pathname*"	Loads the specified scenario. The pathname can be absolute or relative from the *./bin* directory. Mutually exclusive with the `-T` option. For details, see"7.4.2 Loading a Scenario from the Command Line" on page 251.
`--scenarioPerformanceConfig`	Specifies a config file that is written in MTL. It's for internal use in nightly autotests and performance tests. It's here for when we check the command-line options lists.
`--sendBufferSize` *size*	Send buffer size. Default: -1.

Table 11. vrfSim command-line options (continued)

Option	Description
(--sessionId | -i) *ID*	Specifies the session ID for this instance of the GUI. For details, see"4.1.3 Specifying a Session ID" on page 133.
(--settingsFile | -d) *file_name*	Specifies a configuration file to load instead of *vrfSimSettings.xml*.
--setMainThreadToHighPriority	Sets the VR-Forces main thread to high priority. When enabled, other threads may be starved if there are insufficient processors available. If running on hardware with limited processor time, this may decrease overall system performance. (Windows only)
--sharedDataDir *directory_path*	Use this to specify the location of the shared data directory if you installed it somewhere other than the default, which is *./SharedData*.
--sharedDataRoot *directory_root*	Specifies the location of the shared data root. This combined with the stem and data version creates the shared data path.
--sharedDataStem *directory_stem*	Specifies the location of the shared data stem. This combined with the data root and data version creates the shared data path.
--sharedSettingsDirectory *directory*	Specifies the directory where shared settings are stored. The default directory is *$(APP_DIR)/settings*. If the --useUserSettingsDirectory flag is set, then the user settings directory is used to store the settings instead. If not, the application's configuration location (system-dependent) is used.
--simulationModelSet *sms*	Specifies the simulation model set. It must be declared separately for GUI and sim engine, even in combined mode.
(--siteId | -s) *ID*	Specifies the site ID. For details, see "5.4.1 Specifying the Site ID and Application Number" on page 189.
(--startInRunMode | -r)	Starts the sim engine in run mode. If you start the sim engine without specifying a scenario (with -L) or a terrain database (with -T), the sim engine starts in pause mode.

Table 11. vrfSim command-line options (continued)

Option	Description
`--startMinimized`	Minimizes the vrfSim console at startup.
`--subnetMask` *mask*	Specifies a subnet mask. VR-Forces can usually figure out the subnet mask from the IP address. However, if it does not do this correctly, you can explicitly specify the subnet mask using this command-line option.
`--suppressDataAvailableException`	Do not throw an exception when dataAvailable is not checked in non-blocking terrain calls.
`--targetFrameRate` *rate*	Specifies the frame rate above which vrfSim will yield CPU time to other processes.
(`--terrainDatabase` | `-T`) *terrain_database*	Specifies a terrain database to load in the sim engine. Mutually exclusive with the `-L` option. For details, see "3.2.5 Working with Multiple Simulation Engines" on page 118 and "4.5.1 Loading a Terrain Database at Startup" on page 145.
`--testOutputFilename` *filename*	The name of the file to which you want to log scenario autotest outputs.
`--tickPublishersDuringEntityTick`	Tick publishers during the entities' ticks.
`--tracyPort` *integer*	Allows you to specify a sim engine port to profile with the Tracy Client. You can then profile a different sim engine port than the default, and also profile sim engines separately from GUIs. Be sure to include the colon before the port number. The default port is **:10101**.
`--useAbsoluteTimeStamps`	Enables use of absolute timestamps. Otherwise uses relative timestamps.
⚠ If you select `--useAbsoluteTimeStamps`, it is your responsibility to ensure that the time clocks of exercise participants are synchronized. If they are not synchronized, VR-Forces may not work properly.	
`--useAsyncIO`	Use asynchronous I/O. Default: False.
`--useIpv6`	Specifies that the DIS exercise connection use the IPV6 protocol, if available.
`--userDataDir` *directory*	Specifies the location of the user data directory for the sim engine.

Table 11. vrfSim command-line options (continued)

Option	Description
`--useUserSettingsDirectory`	Indicates that the system should store settings in the user login settings directory, rather than the application settings directory. Use in conjunction with `--sharedSettingsDirectory`.
`--waitQueue`	Turn off main thread participation in parallel algorithms.
`(--ignore_rest | --)`	Ignores all command-line options that are listed after this argument. This option is useful for disabling part of a command line in a script for test purposes so that you do not have to retype all of the command options at a later point.
`(--version | -v)`	Displays version information.
`(--help | -h)`	Displays command-line usage information in a console window.
HLA Only	**These are the options specific to vrfSimHLA1516e and vrfSimUserHLA4.**
`--disableRequestUpdates`	Disables the VR-Link Request for updates upon object discovery
`(--execName | -x)` *exec_name*	the name of the federation execution Default: VR-Link.
`(--federateName | -N)` *federate_name*	Specifies a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI automatically assigns a unique name for the federate. Default: VR-Forces. (HLA Evolved only)
`(--federateType | -p)` *type*	Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.
`(--fedFileName | -F)` *fedfile_name*	FED file name.
`--fomMapperInitData` *data*	FOM Mapper initialization data.
`(--fomMapperLib | -f)` *libname*	FOM Mapper library name.
`--fomModules` *module*	Specifies an HLA Evolved or HLA 4 FOM module. Can be used multiple times.
`--ignoreAdvisories`	Enables or disables HLA advisory messages. Default: true.

Table 11. vrfSim command-line options (continued)

Option	Description
(--localSettingsDesignator | -S) *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
--mimModules *HLAE_MIM*	Specifies an HLA Evolved MIM module.
--netnFomRevision *revision_number*	Specifies the internal revision of the implementation of the NETN FOM version as an integer. The default (and currently the only supported revision) is 1.
--netnFomVersion *version_number*	Specifies the NETN FOM version as a double. The default (and currently the only supported version) is 3.0.
--noRtiCompilerCheck	Disable RTI Compiler version check.
--rprFomRevision *revision_number*	RPR FOM revision for RPR FOM 2.0, draft 17.
--rprFomVersion *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
--setFomModuleList	Overwrite FOM Modules with the listed modules. Accepts a comma-separated list of module names; the best practice is to list them in hierarchical order to prevent any issues related to dependencies.
--timeManagement	Enables time management for the sim engine.
DIS Only	**These are the options specific to vrfSimDIS.**
--appIdRange *lower-upper*	Specifies a range of application IDs that represent objects generated by VR-Forces.
--defaultObjectTimeout *timeout*	Specifies the timeout interval for objects.
(--disPort | -P) *port*	DIS port. If you specify a DIS port that is less than 3000 or greater than 4000, discovered objects might not be seen as VR-Forces objects. If you want to use a DIS port outside this range, specify it for the GUI and sim engine with the --appIdRange command-line argument.
--disVersion *version*	The DIS version. Default: 7.
(--exerciseId | -x) *ID*	Exercise ID.

Table 11. vrfSim command-line options (continued)

Option	Description
(--multicastAddresses | -S) *address1 ... addressn*	Specifies one or more multicast addresses.
--suppressSelfReflect	Represses self reflection of the DIS exercise connection.

5.3. Command-Line Options for vrfLauncher

Table 12 describes the command-line options for vrfLauncher (the VR-Forces Launcher).

Table 12. vrfLauncher command-line options

Option	Description
--appDataDir *directory*	Specifies the location of application data. If not specified, VR-Forces uses the default: *./appData*.
(--config | -C)	Opens the Simulation Connections Configuration dialog box, even when a predefined connection is configured for auto-start.
(--connection | -c) *connection_name*	Specifies what predefined connection to use, even if it is different than the one selected as the auto-start option in the GUI.
--dataDir *directory*	Specifies the location of the application's data directory. If not specified, VR-Forces uses the default, *./data*.
(--gui | -g)	Launches the VR-Forces GUI.
(--locale | -G) *language*	Specifies the locale (for localization).
(--run | -r) "*application*"	Launches the specified application, where *application* is the name displayed in the Launcher (case sensitive). It does not need to be selected in the Launcher. If, after the application name, the next argument is additionalargs--, those arguments that follow are added to the command line for the application. The next --run, -runGroup, or --connection stops the additional arguments.
(--runGroup | -R) *group*	Launches all of the applications in the specified application group, regardless of whether they are all currently selected in the Launcher, where *group* is the name displayed in the Launcher (case sensitive). For information on groups, see "4.8. Configuring Applications to Launch" on page 154.
(--sim | -s)	Launches the VR-Forces sim engine.

Table 12. vrfLauncher command-line options (continued)

Option	Description
`--useUserSettingsDirectory`	Indicates that the system should use the user login settings directory instead of the shared application settings.
`(--version | -v)`	Displays version information.
`(--help | -h)`	Displays command-line usage information in a console window.

5.3.1 Running in Combined Mode from the Command Line

If you want to run VR-Forces in combined mode from the command line, there are no command-line arguments for vrfGui or vrfSim to support this. However, you can run vrfLauncher with the `(--connection | -c)` *`connection_name`* argument and VR-Forces will load the specified connection without displaying the Simulation Connections Configuration dialog box. This has the effect of running directly in combined mode from the command line. For example, suppose you want to start in combined mode using the DIS (7) connection. The command would be:

```
vrfLauncher --connection "DIS (7)"
```

This starts all applications that were selected in the VR-Forces Launcher the last time you ran VR-Forces. If you want to ensure that specific applications run, use the `(--run | -r)` "*`application`*" command-line option for each application you want, where *`application`* is the name as it displays in the Launcher window. You can specify additional options for the application, as described for each of the applications. (See "5.3.2 Using the additionalargs-- Command-line Argument" below.)

If a predefined connection has spaces in the name, you must surround the name with quotes, for example, `"DIS (7)"` or `"HLA 4"`. You must launch a predefined connection from the vrfLauncher at least once so that VR-Forces can save the network address information it requires to launch.

5.3.2 Using the `additionalargs--` Command-line Argument

The `additionalargs--` command-line argument is used to send additional command-line arguments to applications that you run with `vrfLauncher`.

For example, in this command line argument:

```
vrfLauncher --connection "DIS (7)" --run "VR-Forces GUI"
additionalargs-- --showConsole --run "VR-Forces Simulation Engine"
additionalargs-- --notifyLevel 3
```

- `vrfLauncher --connection "DIS (7)"` uses the predefined connection for DIS (7).
- `--run "VR-Forces GUI" additionalargs-- --showConsole` runs the GUI with the console.

- `--run "VR-Forces Simulation Engine" additionalargs--`
  `--notifyLevel 3` sets the sim engine's notification level to 3 (verbose).

5.4. Protocol-Independent Command-line Options

This section describes in greater detail some of the most commonly used command-line options that you can use with both DIS and HLA.

5.4.1 Specifying the Site ID and Application Number

Each GUI and sim engine must have a unique site ID:application number pair. To specify the site ID, use `(--siteId | -s)` *`ID`*. To specify the application number, use `(--appNumber | -a)` *number*, for example:

```
vrfSimDIS --siteId 2 --appNumber 3005
```

The default application number for vrfGui is 3101; for vrfSim it is 3001.

5.4.2 Specifying the Language to Use in the GUI

VR-Forces decides which translation files to use based on the language setting for your computer. However, you can override the language setting with the (--locale | -G) *language* command-line option, for example, this command sets the language to German:

```
vrfGui --siteId 1 --appNumber 3000 --locale de --dis
```

If you use this command-line option, you must have a file named *vrf_language_code* (for example, *vrf_de*) in the *./translations* directory. For more information, see "2.6. Localizing the Graphical User Interface" on page 108.

5.4.3 Setting the Notification Level

You can control the types of messages sent to the console (and log file on a PC) by specifying a notification level.

➤ To specify a notification level, use the `(--notifyLevel | -n)` *`notification_level`* option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --notifyLevel 1
```

The notification levels you can specify are:

- 0 – Only fatal messages print.
- 1 – Warning messages and fatal messages print.
- 2 – Some diagnostic information prints with warnings and fatal messages.
- 3 – Verbose information prints.
- 4 – Debug information prints.

The default notification level is 2.

> The Linux version of VR-Forces does not have a log file. Informational messages are sent to the window from which you start VR-Forces. To capture informational messages, redirect output to a file.

5.4.4 Loading Plug-ins

You can load plug-ins from the command line or by specifying them in a configuration file. (For details about creating a configuration file, see "4.9. Managing Plug-ins" on page 156.)

➤ To load a plug-in from the command line, use the `--loadPlugin` *`plugin_filename`* command-line option, for example:

```
vrfGui --siteId 1 --appNumber 3000 --loadPlugin pluginA
  --loadPlugin pluginB --dis
vrfSimDIS --siteId 1 --appNumber 3001 --loadPlugin pluginA
   --loadPlugin pluginB
```

> Do not specify the extension of the plug-in (for example, *.dll*). If the extension is included, the plug-in will not load.

> If your plug-ins are not in the *./appData* directory of the VR-Forces installation, you can use the `--appDataDir` command-line option to point to the directory where your plug-ins are located, for example:
>
> `--appDataDir D:/sharedDevelopment/plugins`

VR-Forces can display a list of the plug-ins that are loaded. For details, see "4.9.7 Viewing a List of Loaded Plug-ins" on page 162.

5.5. Command-Line Options for HLA Federations

VR-Forces listens to whatever port is specified in the RID file. For information about the RID file, consult your RTI documentation.

5.5.1 Specifying the Federation Execution

By default, VR-Forces uses the *MAK-RPR2-1-1.fed* file and the MAK-RPR-2.0 federation execution.

➤ To specify the HLA federation execution name, use the `(--execName | -x)` *`exec_name`* option, for example:

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --execName MAK-RPR-2.0
   --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed
```

> - If you are using the MAK RTI and are running multiple, concurrent federation executions, you must run the rtiexec.
> - You cannot use the MAK RTI in lightweight mode with VR-Forces.

5.5.2 Specifying the FED File

In combined mode, specifying `-F` in the command line also applies to the sim engine. If you are running a sim engine independently and want to specify a FED file name, you must use (`--fedFileName | -F`) *`fedfile_name`*.

➤ To specify a FED file at startup, use the `--fedFileName` option.

5.5.3 Specifying the RPR FOM Version

VR-Forces has built-in support for the RPR FOM. The default version is 2.0.

You can specify a different RPR FOM version (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0). If you specify a different FOM version, you must also specify the correct FED file (with `--fedFileName`).

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --rprFomVersion 2.0
   --fedFileName MAK-RPR2-1-1.fed
```

ⓘ Please see the Product Specifications & Dependencies (https://www.mak.com/mak-one-specs) for a list of the RPR FOM versions supported by your version of VR-Forces.

5.5.4 Specifying a FOM Mapper

VR-Forces supports alternative FOMs through the VR-Link FOM Mapper. You can specify which FOM Mapper to use with the (`--fomMapperLib | -f`) *`libname`* command-line option:

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --execName MAK-RPR-2.0
  --rprFomVersion 2.0 -F MAK-RPR2-1-1.fed --fomMapperLib myFomMapper
```

5.5.5 Specifying the RPR FOM Revision

MAK revises FED files when it discovers errors in how it has previously handled various aspects of the HLA standard. It maintains multiple revisions to ensure compatibility with applications that depend on previous revisions.

➤ To specify the RPR FOM revision, use the `--rprFomRevision` *`revision`* command-line option, for example:

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --rprFomVersion 2.0
  --fedFileName MAK-RPR2-1-1.fed --rprFomRevision 2
```

5.5.6 Specifying FOM Mapper Initialization Data

Some FOM Mappers require initialization data. If you specify a FOM Mapper that requires such data, specify the data with the `--fomMapperInitData` option:

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --execName MAK-RPR-2.0
  --rprFomVersion 2.0 --fedFileName MAK-RPR2-1-1.fed
  --fomMapperLib myFomMapper --fomMapperInitData ABC
```

The string "ABC" is passed to the FOM Mapper creation function in the shared library as an initialization parameter.

5.5.7 Specifying a FED File That Is Appropriate for the FOM Mapper

> ⚠ The (`--fomMapperLib | -f`) *`libname`* option simply tells the application which FOM Mapper to use. It does not ensure that you are using the correct FED file for the FOM Mapper. Make sure that you are using a FED file that is consistent with the FOM Mapper you have chosen.

You can specify a FED file using the (`--fedFileName | -F`) *`fedfile_name`* option.

This example instructs the application to use the RPR FOM 2.0, version 17 FOM mapping code, in the federation execution MAK-RPR20017-1-1, with the *MAK-RPR20017-1-1.fed* file:

```
vrfSimHLA1516e --siteId 1 --appNumber 3001 --rprFomVersion 2.0017
  --execName MAK-RPR20017-1-1 --fedFileName MAK-RPR20017-1-1.fed
```

5.5.8 Enabling Time Management

You can enable time management for the sim engine from the command line. See "Enabling Time Management for the Simulation Engine" on page 144 for details.

5.6. Command-Line Options for DIS Exercises

This section describes command-line options that apply only to using VR-Forces with DIS exercises.

5.6.1 Specifying the Port Number

By default, VR-Forces listens to port 3000.

If you specify a DIS port that is less than 3000 or greater than 4000, discovered objects might not be seen as VR-Forces objects. If you want to use a DIS port outside this range, specify it for the GUI and sim engine with the `--appIdRange` *`lower-upper`* command-line argument.

➤ To change the port that VR-Forces listens to, use the (`--disPort | -P`) *`port`* option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --disPort 3500
vrfGui --siteId 1 --appNumber 3000 --dis --disPort 3500
```

5.6.2 Specifying Point-to-Point or Multicast Operation

You can specify that VR-Forces run in point-to-point mode or multicast.

➤ To specify point-to-point, use the (`--destAddrString | -A`) *`address`* option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --destAddrString 123.123.12.1
```

➤ To specify multicast, use the (`--multicastAddresses | -S`) *`address1 ... addressn`* option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 -S 233.233.22.3
```

➤ To specify more than one multicast address, use the `--multicastAddresses` option multiple times, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --multicastAddresses address1 --
multicastAddresses address2 ...
```

The default is broadcast mode.

5.6.3 Specifying the Multicast Time-to-Live

The multicast time-to-live specifies the number of routers that a message can pass through when PDUs are sent to multicast addresses.

➤ To specify the multicast time-to-live value, use the `--mcastTtl` option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --multicastAddresses address --
multicastAddresses address --mcastTtl 3
```

5.6.4 Using Asynchronous I/O

If you want to use asynchronous I/O for DIS exercises, start VR-Forces with the `--useAsyncIO` option. Using asynchronous I/O can help prevent dropped packets and the resulting loss of information. There is no comparable command-line option for HLA. To use asynchronous I/O in HLA, you set a RID file parameter.

5.6.5 Specifying the Exercise ID

➤ To set the VR-Forces exercise ID, use the `(--exerciseId | -x)` *`ID`* option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --exerciseId 6
```

5.6.6 Specifying the DIS Version

➤ To specify the DIS version, use the `--disVersion` *`version`* command-line options, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --disVersion 7
```

See "1.6. Distributed Simulation Standards Supported" on page 93 for a list of supported versions.



6. Optimizing Performance

This chapter explains how to monitor and improve performance.

6.1. Introduction 196
6.1.1 Optimizing Simulation Engine Performance 196
6.1.2 Optimizing GUI Performance 199
6.2. Monitoring the Simulation Engine Performance 200
6.2.1 Using the Tracy Profiler in the VR-Forces Simulation Engine 200
6.3. Monitoring Script Frame Rates 201
6.4. Displaying Performance Statistics 202
6.4.1 Displaying VR-Forces Performance Statistics 202
6.4.2 Using the Tracy Client 203
6.5. Clearing the Model Instancing Cache 204
6.6. Miscellaneous Performance Configuration Options 205
6.6.1 Limiting Use of Spot Reports 205
6.6.2 Using Asynchronous I/O for DIS Exercises 206
6.6.3 Using Incremental Compiling with Streamed Data 206
6.7. Setting the Tick Rate 206
6.7.1 Specifying the Tick Rate for Components 207
6.8. Configuring Graphics Quality 207
6.8.1 Alpha to Coverage 207
6.8.2 Using the Ambient Occlusion Effect 209
6.8.3 Configuring Ambient Occlusion 211
6.9. Balancing Visual Quality Against Network Performance 212
6.9.1 Enabling Configuration of Performance Settings 213
6.10. Trajectory Smoothing 213
6.10.1 Configuring Trajectory Smoothing 214
6.10.2 Configuring Articulated Part Smoothing 215
6.11. Tuning the Target Frame Rate 215
6.12. Coloring Draw Calls 215

6.13. Configuring VSync 216
6.13.1 Managing 3D Settings and Multiscreen Performance 217
6.14. Adaptive Tick Rate 217
6.15. Preloading Models 218
6.15.1 Command-Line Options for Preloading Models 219
6.16. DI-Guy Performance Settings 220
6.17. Specifying Terrain Categories 221
6.18. Using the Developer Debug Overlay 221
6.19. Using the MAK Remote Debugger 223

6.1. Introduction

VR-Forces's overall performance varies based on these factors:

- Size of the terrain database.
- Amount of memory available on your computer.
- Computer speed and processing power.
- The number of objects (simulation objects and tactical graphics) in the scenario and how active the simulation objects are.
- The number of objects you try to simulate on a given sim engine.
- The frequency of the tick rate.
- The number of GUIs and sim engines you run on your computer.



- If you experience poor performance, consider changing some settings. See "A.2.1 Features You Can Disable to Improve Performance" on page 1660.
- Nahimic audio software can have a negative impact on performance. See "A.2. Performance and Display Issues" on page 1660 to learn how to add the application to the *BlackApps.dat* file.
- On Windows, the Fault Tolerant Heap (FTH) can also affect performance. See "A.3. The Fault Tolerant Heap" on page 1661.

6.1.1 Optimizing Simulation Engine Performance

VR-Forces supports these methods for optimizing the performance of the simulation engine:

- Adjusting the refresh rates. (For details, see "6.9. Balancing Visual Quality Against Network Performance" on page 212.)
- Turning off processor-intensive features. (For details, see "6.6. Miscellaneous Performance Configuration Options" on page 205.)
- Specifying the VR-Forces sim engine sleep time. (For details, see "6.11. Tuning the Target Frame Rate" on page 215.)

- Optimizing the terrain. (For details, see the various chapters in the "Section XII. Terrain" on page 1237 section, particularly "Chapter 63. Introduction to Terrain Usage in VR-Forces" on page 1241, and also the *Terrain Performance and Configuration* chapter in the *Adding Content to MAK ONE Applications Reference Manual*.)

Optimizing Multiprocessor Usage for Simulation Engine

The VR-Forces simulation engine is a multithreaded application that you can configure to make optimal use of as many CPUs as the hardware can make available. The default configuration of VR-Forces limits the overall CPU usage. You should make adjustments to take advantage of your specific hardware setup and optimize performance on large scenarios. Use the Performance Monitor panel and Tracy to understand the performance characteristics as you adjust the values. See "6.2. Monitoring the Simulation Engine Performance" on page 200 and "6.4.2 Using the Tracy Client" on page 203.

To adjust these settings, edit the *./appData/settings/vrfSim/vrfSim.mtl* file.

- numCallbackThreads. The number of additional threads that are spawned to run simulation ticks on locally simulated objects and to perform various other simulation activities each frame. Increase this number if the Object Tick section of the frame is greater than the Network Tick section in the Performance Monitor. Scenarios with higher complexity entities typically can be improved by increasing the callback thread count. You can adjust this value at runtime using the # Threads field in the Performance Monitor panel.
- numNetworkCallbackThreads. The number of threads to send and process network updates. Increase this number if the Network Tick section of the frame is greater than the Object Tick section in the Performance Monitor. Scenarios with many low complexity entities and/or many remote entities can typically be improved by increasing the network callback thread count.

  > Using a number greater than 1 for this setting can cause instability if the RTI is not thread safe. The default configuration of VR-Forces is to force this value to 1 if VR-Forces cannot determine that it is safe to use multiple. See "Network Connections and Multithreading" on the next page for details.

- numberOfNavQueueProcessingThreads. The number of threads to use for navigation path queries when entities are using the navigation mesh. Increasing this value will typically path planning entities more responsive (start moving along a new path more quickly) but will not change the simulation frame rate.

For best performance on a heavily loaded simulation engine, be sure that the sum of these three thread counts is about 2 less than the total available logical CPUs on the hardware. The terrain paging thread and the main thread are in addition to these threads. If the VR-Forces GUI is going to run on the same hardware as VR-Forces Simulation Engine, you should leave at least 2 CPUs should be left free for that purpose.

> In VR-Forces 4.10 and earlier, MAK recommended that users run multiple simulation engines on the same computer to take advantage of multiprocessing capabilities. With VR-Forces 5.0 and later, you should run only one simulation engine per computer, with the above configuration changes made to use as many CPUs as desired. Running multiple simulation engines on the same computer is not optimal and is no longer recommended.
>
> If you find that scenario performance does not meet your needs after optimizing multiprocessor usage, you might want to use multiple machines and multiple sim engines. See "7.5. Load Balancing Scenarios" on page 255 for more information.

Network Connections and Multithreading

For scenarios with large numbers of entities and/or frequent entity updates, the HLA connections are recommended.

When using an HLA connection, the RTI in use as well as the configuration may or may not be compatible with multithreaded calls. If the RTI is not compatible with multithreaded calls and VR-Forces is using more than one network thread, the application becomes unstable. To help ensure stable behavior, VR-Forces checks the RTI that is being used and allows only multiple network threads if it recognizes the RTI and is able to set it to a multiprocessing mode.

Using the MAK RTI:

- MAK RTI 4.6b and newer. VR-Forces sets Processing Model to 2 automatically. You do not need to make any additional configuration to get multiple network threads.
- MAK RTI 4.6a and earlier. The VR-Forces default configuration allows only one network thread. To get multiple network threads:
    1. Set RTI_*processingModel* to 2 in the RTI *rid.mtl* file.
    2. Set useSingleNetworkThreadWhenRtiThreadSafetyNotGuaranteed to 0 in *vrfSim.mtl*.

Using other RTIs: The VR-Forces default configuration allows only one network thread. To get multiple network threads:

1. Configure your RTI if needed for multithreaded calls.
2. Set useSingleNetworkThreadWhenRtiThreadSafetyNotGuaranteed to 0 in *vrfSim.mtl*.

Optimizing HLA Object Discovery

In large scenarios of approximately 10,000 published entities or more, the startup of the scenario can cause a large spike in network traffic and a long delay when all the entities are created at approximately the same time. The more federates there are participating in the exercise, the worse the problem becomes. This is due to a large number of HLA Request Attribute Value Update service calls performed in succession.

VR-Forces has some configuration that helps reduce this by delaying Request Attribute Value Update service calls until after all the entities are created and have had their initial state sent out when loading a scenario. To enable this, use the command line argument `--requestWhenInitialized` when launching vrfSim and vrfGui.

6.1.2 Optimizing GUI Performance

The performance of the GUI depends on:

- Processor speed.
- Video card. (Always use the latest drivers for your video card.)
- Available memory.
- Size of the terrain database.
- The number and complexity of models loaded and how well they have been optimized for performance. For information about optimizing models, see "Best Practices for Creating Models" in the *Adding Content to MAK ONE Applications Reference Manual*.
- Number of objects in the simulation and how active they are.
- Use of model instancing.

You can display statistics to help you evaluate VR-Forces's performance. For details, see "6.4. Displaying Performance Statistics" on page 202.

These display features can affect performance:

- Displaying performance statistics.
- Use of ground clamping.
- Displaying track histories.
- Displaying height-above-terrain lines.
- Displaying radio communications lines.
- Enabling advanced lighting and various lighting effects.
- Enabling dynamic ocean effects. When you are using Plan View observer mode, enabling wakes, spray effects, and the buoyancy model can seriously degrade performance. These features are disabled by default. If you inadvertently enable them and experience this problem, switching to Stealth mode and back to Plan View mode restores performance.
- Sound effects. Disabling sound effects can improve performance.
- You can configure anti-aliasing settings and Horizon Based Ambient Occlusion (HBAO) to improve performance. For details, see "6.8. Configuring Graphics Quality" on page 207.

In addition to the sections in this chapter and in *Terrain Performance and Configuration* in the *Adding Content to MAK ONE Applications Reference Manual*, the following sections describe options for optimizing GUI performance:

- "Configuring SpeedTree Trees" in the *Adding Content to MAK ONE Applications Reference Manual*.
- "6.5. Clearing the Model Instancing Cache" on page 204.
- "83.4.1 Setting the Clipping Planes" on page 1605.

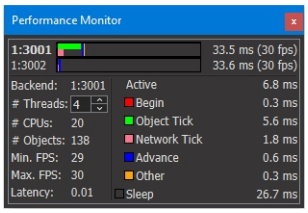
- "Setting the Ocean LOD Elevation" in the *Adding Content to MAK ONE Applications Reference Manual*.
- "Configuring the Ocean Height Map" in the *Adding Content to MAK ONE Applications Reference Manual*.
- "53.2. Displaying Wakes and Spray Effects" on page 1110.
- "52.7. Configuring the Skybox Cube Map" on page 1096.
- "61.1.9 Improving Performance when Attaching to Ground Objects" on page 1214.
- "53.3. Enabling Buoyancy for Surface Entities" on page 1111.
- "Chapter 52. Lighting Effects" on page 1087.

6.2. Monitoring the Simulation Engine Performance

You can monitor performance statistics for each sim engine in your session using the Performance Monitor Panel (Figure 36). At the top, it shows the frame rate statistics for each sim engine. You can also see the details for a sim engine by selecting it.

For details about the information in the Performance Monitor Panel, see "VR-Forces Simulation Engine Architecture" in the *VR-Forces Developers Guide*.

Figure 36. Performance Monitor



➤ To view the Performance Monitor Panel, choose **View > Performance Monitor Panel**.

6.2.1 Using the Tracy Profiler in the VR-Forces Simulation Engine

You can get more performance statistics from the simulation engine using the Tracy profiler.

➤ **To use the Tracy profiler:**

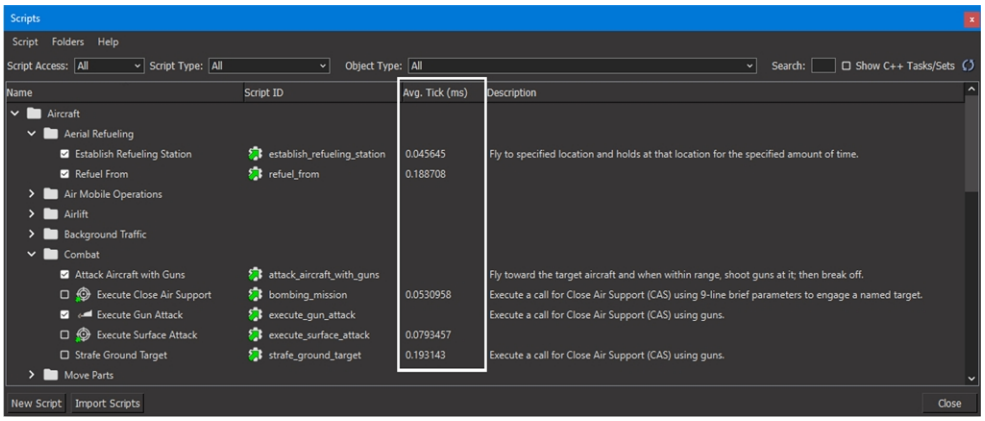
1. Start VR-Forces normally.
2. Start the Tracy Client. (The Tracy Client is in Tools shortcuts folder, which accessible from the VR-Forces Start menu, or in the *./bin64* directory).
3. In the VR-Forces Sim Engine console, enter "tracy enable" and then press **Enter**.
4. Load a VR-Forces scenario and run it.
5. The vrfSim should show on the Tracy Client window. Double-click it to attach and gather statistics.

For more information, see "6.4.2 Using the Tracy Client" on page 203.

6.3. Monitoring Script Frame Rates

VR-Forces can display the average time per tick for scripted tasks and other scripts. When this is enabled, the Scripts window shows the frame rate for all active scripts (Figure 37).

Figure 37. Script frame rate

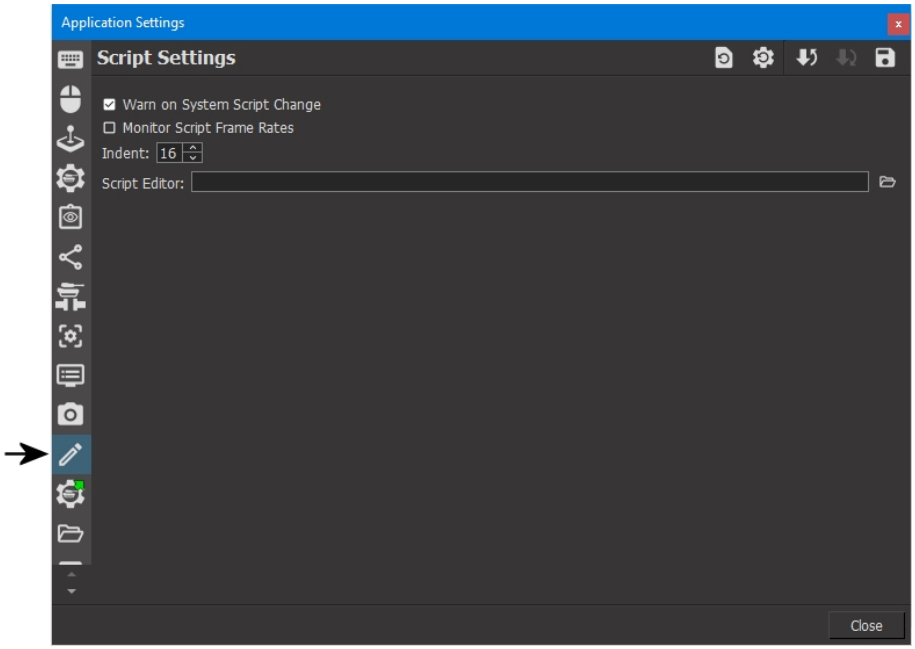


For information about scripts and the scripts window, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743

➤ **To display the frame rate for scripts:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 38).

Figure 38. Script Settings page



3. Select Monitor Script Frame Rates.

4. Choose **Simulation > Scripts**. The Scripts window opens (Figure 37). A column called Avg Tick is added. It displays the tick rate for active scripts.

You can also open the Script Settings page from the Scripts menu in the Scripts dialog box.

6.4. Displaying Performance Statistics

VR-Forces can display VR-Forces performance statistics and a function profiler (Figure 39).

Figure 39. VR-Forces performance statistics and function profiler



Displaying performance statistics can affect performance.

6.4.1 Displaying VR-Forces Performance Statistics

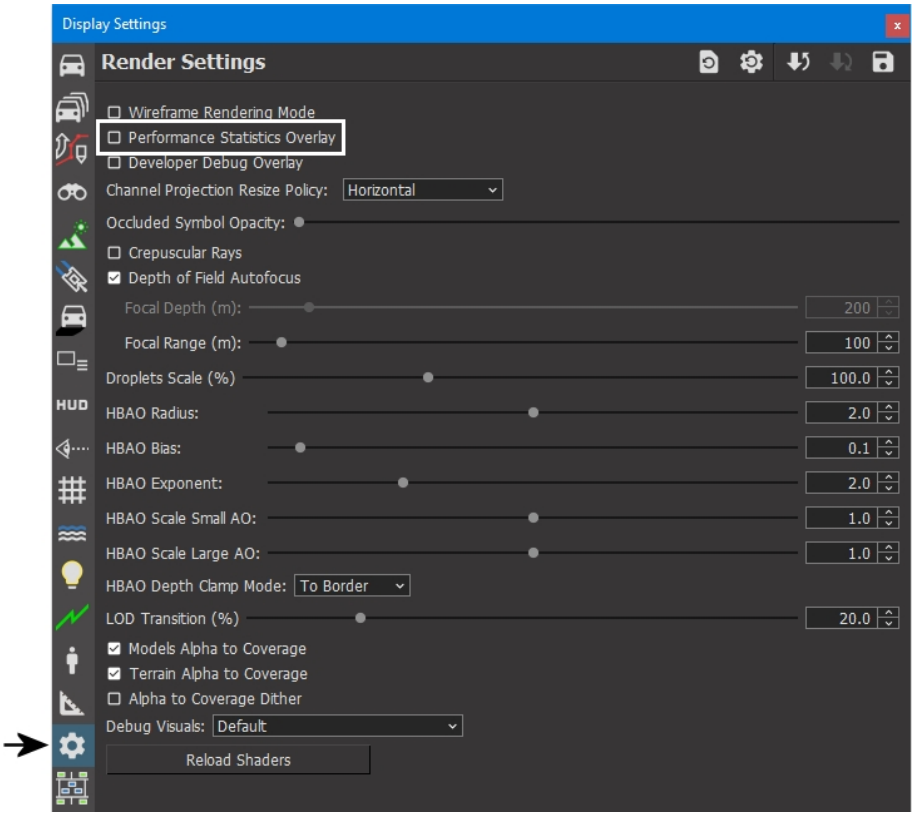
The VR-Forces Performance Statistics shows information about the scene, while the Function Profiler is primarily for the use of developers who are debugging VR-Forces applications. For information about how to integrate the profiler into your code, please see Chapter 10, *Using the Function Profiler*, in the *Vantage Developers Guide* section of the *VR-Forces Developers Guide*.

For information about another debugging utility, see "6.18. Using the Developer Debug Overlay" on page 221.

➤ **To display VR-Forces performance statistics:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40).

Figure 40. Render Settings page



3. Select the Performance Statistics Overlay check box. A statistics display is added to the window (Figure 39 on the previous page).

6.4.2 Using the Tracy Client

VR-Forces includes the Tracy Client, which you can use to evaluate performance or save a file to send to MAK Support to help debug performance issues. To learn more about using the Tracy Client, read the documentation at https://bitbucket.org/wolfpld/tracy/downloads.

⚠ VR-Forces is compatible with the Tracy Client version that was included in the installation package. If you update Tracy Client, it might no longer be compatible.

➤ **To run the Tracy Client and save a file:**

1. Choose **Settings > Display**. Display Settings dialog box opens. Select the Render Settings page (Figure 40 above).

2. Select the Performance Statistics Overlay check box. A statistics display is added to the window (Figure 39 on page 202).
3. Next to Collect Stats, select the Tracy check box.
4. To run Tracy Client:
   - On Windows, from the Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Tracy shortcut.
   - On Linux, run *./bin64/Tracy*.
5. Verify the client address and then click Connect. The Tracy Client displays the information from VR-Forces.
6. To save the file, click Connection (). The Connection dialog opens.
7. Click Save Trace.
8. Save the file.

> You can also open the Performance Statistics Overlay by pressing **F2** twice and then selecting the Tracy check box next to Collect Stats.

6.5. Clearing the Model Instancing Cache

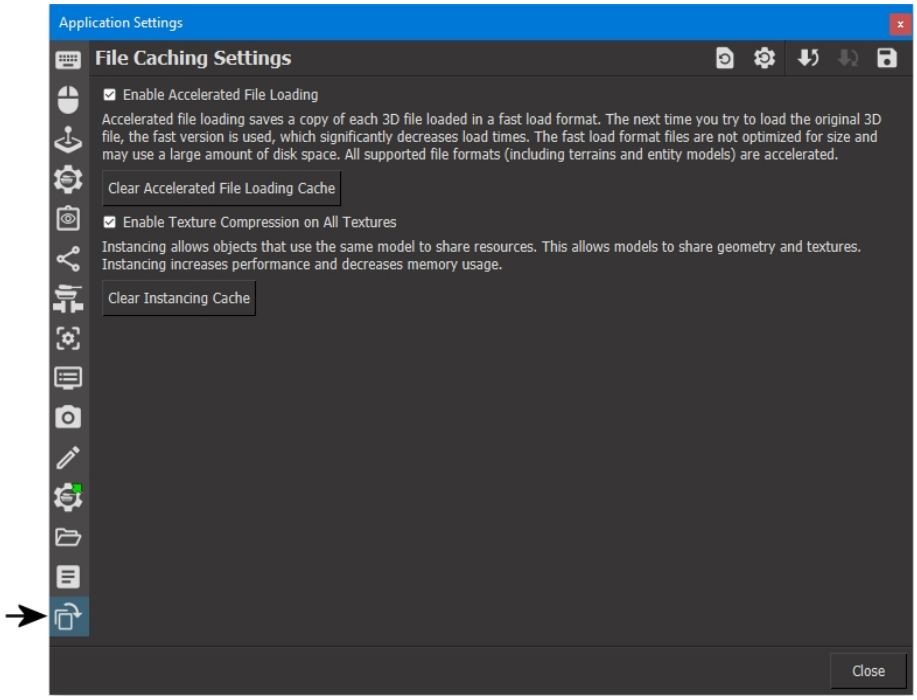
VR-Forces has a feature called model instancing that lets models share resources, such as geometry and textures. These shared resources are stored in a cache. Use of model instance caching decreases memory use and improves performance.

You can clear the model instancing cache if you need the disk space.

➤ **To clear the model instancing cache after each scene change:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 41).

Figure 41. File Caching Settings page



3. Select the Clear Instances After New Scene check box.

➤ **To clear the model instancing cache manually:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the File Caching Settings page (Figure 41).
3. Click Clear Instancing Cache.

6.6. Miscellaneous Performance Configuration Options

You can improve GUI performance by limiting use of display features:

- Turn off track histories.
- Limit use of object transparency.
- Limit use of spot reports.
- In DIS, use asynchronous I/O.
- For streamed data, use incremental compiling.

6.6.1 Limiting Use of Spot Reports

Use of spot reports can degrade performance. You may want to limit the number of simulation objects for which they are generated or disable spot reports entirely. You can also specify that spot reports be sent only to the GUI. For details about using spot reports, see "9.1. Displaying Spot-Reported Simulation Objects" on page 290.

6.6.2 Using Asynchronous I/O for DIS Exercises

By its nature, DIS generates a lot of network traffic. Therefore, you may experience problems with dropped packets and simulation objects may time out. You can mitigate this problem by using asynchronous I/O. To use asynchronous I/O, start with the `--useAsyncIO` command-line option or select Use Asynchronous I/O in the Simulation Connections Configuration dialog box.

6.6.3 Using Incremental Compiling with Streamed Data

OSG option that allows the processing of streamed data to be spread across multiple frames, reducing performance spikes. To use it, create the environment variable OSG_ICO. (It does not need a value. It just needs to exist.)

> Using OSG_ICO may cause LOD issues with other paged formats.

6.7. Setting the Tick Rate

VR-Forces lets you tune performance by varying the tick rate of components, state repositories, and network interfaces. By decreasing the frequency with which some simulation object components are ticked, you make more CPU time available to other simulation objects. For example you could decrease the tick rate for slow moving entities such as dismounted infantry or command posts, making more CPU available for entities such as aircraft or missiles.

Variable tick rates are determined by a minimum tick period and a minimum tick period variance. The minimum tick period parameter specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's components. VR-Forces may tick the components less frequently (due to heavy CPU load), but it will never tick the entity's components more frequently than specified.

To reduce the frequency that a component can be ticked, increase the minimum tick period value. To increase the frequency that a component can be ticked, reduce the minimum tick period value.

The minimum tick period variance argument specifies an amount of variance allowed around the minimum tick period. Specifying a minimum tick period variance value helps prevent the spikes in CPU usage that would result if all components tried to tick at the same time. Calculation of the time to tick a component using minimum tick period variance is as follows (the example uses the entity component arguments):

If the current time is t_0 and the next scheduled tick is t_{next} then:

$$t_{next} = t_0 + min\text{-}tick\text{-}period +/- rand(min\text{-}tick\text{-}period\text{-}variance/2)$$

If you insert some sample values (in seconds), you get:

$$t_{next} = .1 + .05 +/- rand(.05/2)$$

This means that the next tick will be no sooner than the current time plus .15 seconds plus or minus a random value between 0 and .0125 seconds.

> If the next tick time calculated by this formula is less than the current tick time, it is set to the current tick time, which results in the component being ticked on the next tick.

6.7.1 Specifying the Tick Rate for Components

In the object parameter database, you can specify a **min-tick-period** and **min-tick-periodvariance** for each simulation object and for individual simulation object components. The component-level settings, if any, override the simulation object-level settings.

6.8. Configuring Graphics Quality

VR-Forces has command-line and display settings options that let you configure graphics quality. In particular, depending on the capabilities of your graphics card, you may need to adjust the default settings for anti-aliasing using `--antiAliasing` *level*. Anti-aliasing is a technique used to reduce the jagged edges of digital graphics. You can set the level to 0, 2, 4, 8, or 16. Default: 4. On older graphics cards, you may need to set anti-aliasing to 0 to improve performance.

You can also make adjustments to the horizon-based ambient occlusion. See "6.8.3 Configuring Ambient Occlusion" on page 211.

6.8.1 Alpha to Coverage

Alpha to coverage is a way of handling objects with transparency that improves rendering of objects that have a sharp transition between opaque and transparent. Grass, trees, and fences are a few examples of when alpha to coverage is a good option, as shown in Figure 42. In addition to improving visualization of some objects, there is a slight performance improvement when alpha to coverage is enabled.

Figure 42. Alpha to coverage - on and off



You can set alpha to coverage for models and terrains. For models, you can enable it globally or in specific model definitions. For terrains, you can enable it globally or for specific terrain patches.

By default, alpha to coverage is enabled globally for models and terrains.

If alpha to coverage is enabled, you can enable a dither shader that modifies the alpha with a noise pattern so that there are no hard edges between different alpha levels. The tradeoff for this setting is that it adds a small amount of noise as the object or camera moves.

VR-Forces uses render bin 1 for the alpha to coverage render bin. You can change the render bin with the VRV_ALPHA_TO_COVERAGE_RENDERBIN environment variable. (Render bins are an OpenSceneGraph concept that affects the order in which scene objects are rendered. Do not set the render bin environment variable unless instructed to do so by a plug-in developer.)

Enabling Alpha to Coverage for Models

You can enable alpha to coverage globally for models and you can specify it for specific models. When you enable alpha to coverage globally, it affects all models that do not have it specified specifically in their model definition.

To enable or disable alpha to coverage for a specific model, add the alphaToCoverageEnabled parameter to the model definition and set it to true or false. For information about editing model definitions, see "Editing a Model Definition" in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To enable or disable alpha to coverage globally for models:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40 on page 203).
3. Select or clear the Models Alpha to Coverage check box.

Enabling Alpha to Coverage for Terrains

You can enable alpha to coverage globally for terrains or for specific terrain patches. The global setting affects all terrain patches that do not have an alpha to coverage option specified in their MTF file.

You enable alpha to coverage for a terrain patch when you add it. For details, see "Adding Elevation Data to a Terrain" in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To enable or disable alpha to coverage globally for terrains:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40 on page 203).
3. Select or clear the Terrain Alpha to Coverage check box.

Enabling the Dither Shader for Alpha to Coverage

If alpha to coverage is enabled, you can enable a dither shader that modifies the alpha with a noise pattern so that there are no hard edges between different alpha levels. The trade-off for this setting is that it adds a small amount of noise as the object or camera moves.

➤ **To enable or disable the dither shader for alpha to coverage:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40 on page 203).
3. Select or clear the Alpha to Coverage Dither check box.

6.8.2 Using the Ambient Occlusion Effect

Horizon-based ambient occlusion (HBAO) adds screen space ambient occlusion lighting effects to the scene. This improves the realism by approximating the amount of ambient light that impacts a scene, which provides more realistic lighting of three-dimensional objects.

HBAO is disabled in Figure 43 and enabled in Figure 44. The differences are subtle. Some differences you might notice are the light inside the building as seen through the window on the right side of that building, the lighting on the food in the stands, and the shadows on the wall of the building behind the stands and grass.

Figure 43. HBAO disabled



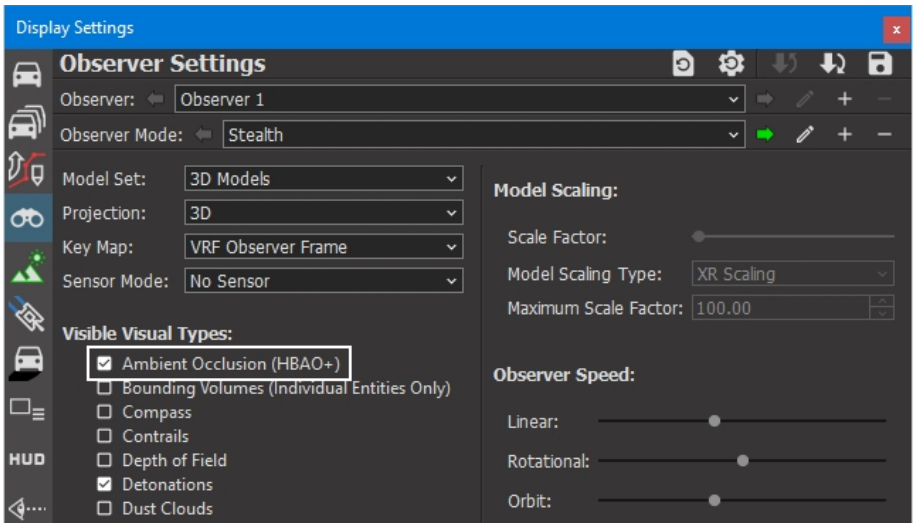
Figure 44. HBAO enabled



➤ **To enable or disable HBAO:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 45).

Figure 45. Observer Settings page



3. Select the 3D observer mode for which you want to enable HBAO.
4. Select or clear the Ambient Occlusion (HBAO+) check box.

6.8.3 Configuring Ambient Occlusion

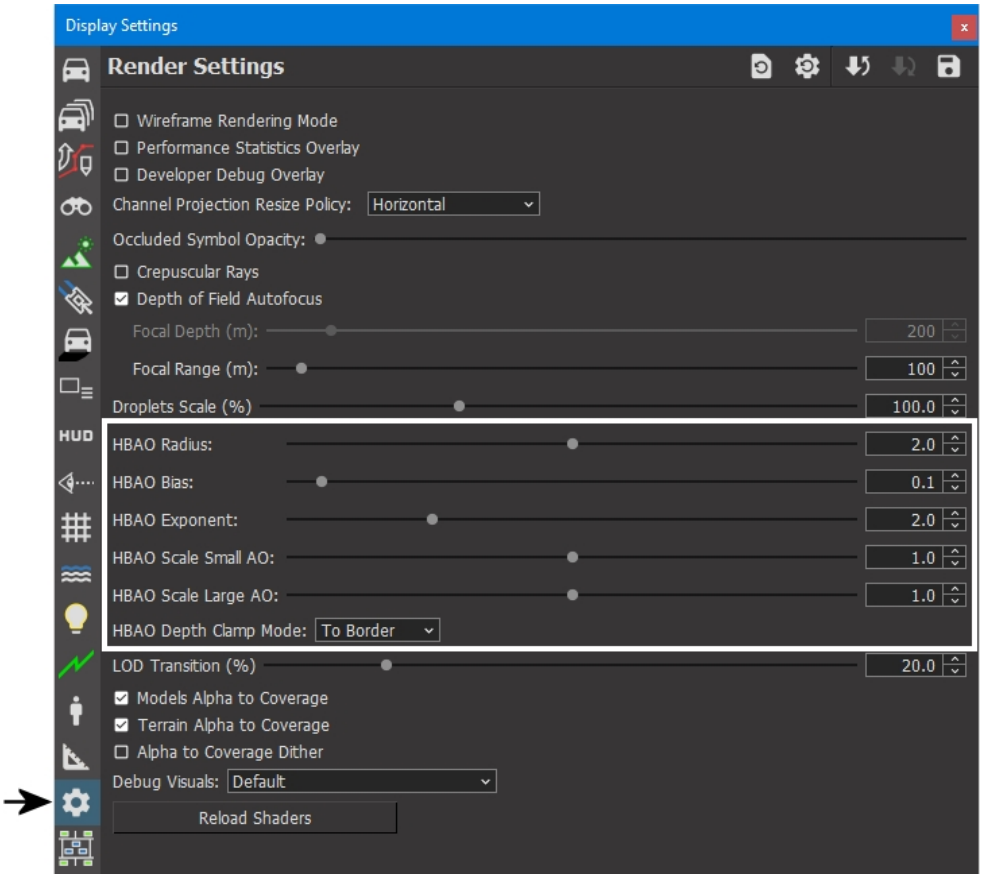
There are parameters that adjust how ambient occlusion appears.

- HBAO Radius. The distance from occluding objects where the HBAO effect is visible. Higher values cause the effect to extend further from occluding objects.
- HBAO Bias. The bias for thin items. At higher values, larger objects cause the most noticeable effect on surrounding objects. This can reduce polygon edge artifacts and artifacts caused by thin geometry.
- HBAO Exponent. The intensity of the HBAO effect. Higher values create a more intense effect.
- HBAO Scale Small AO. The intensity of the HBAO effect on objects that are close to each other. Higher values have a more intense effect on the small corner areas of objects and also objects that are close together.
- HBAO Scale Large AO. The intensity of the HBAO effect on objects that are at the edges of the specified HBAO Radius. Higher values make the area for objects that are around the HBAO radius distance from each other show a more intense effect.
- HBAO Depth Clamp Mode. The depth clamp mode to use. To Border uses a method that reduces the false occlusion of objects along the edge of the screen. To Edge uses a method that correctly applies the occlusion effect to objects along the edge of the screen.

➤ **To configure HBAO:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 46).
3. Adjust the HBAO settings as needed.

Figure 46. Render Settings page



6.9. Balancing Visual Quality Against Network Performance

VR-Forces allows you to tune performance by adjusting dead-reckoning thresholds. The primary benefit of this feature is to improve the appearance of entities in 3D visualization programs such as VR-Vantage Stealth and the VR-Forces Stealth observer mode. Reducing thresholds improves the accuracy of entity movement and reduces jumpy movement. However it does so at the expense of network performance, because a lower dead-reckoning threshold results in more packets being sent for each entity.

When you change thresholds at runtime, VR-Forces sends a message to all sim engines in the scenario to change their default thresholds for translation and rotation. The new thresholds remain in effect until the sim engines are closed. They apply to any scenario run during this time and they override thresholds set in *./appData/settings/vrfSim/vrfSimSettings.xml*. Changes to the default thresholds are saved in *vrfSimSettings.xml*.

If your exercise sends periodic updates, such as a DIS heartbeat, the thresholds evaluate changes since the most recent heartbeat.

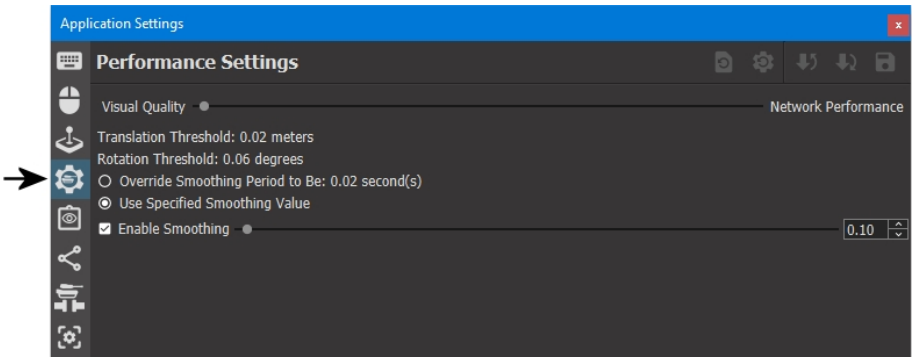
Changes made at runtime have no effect in these cases:

- If a sim engine joins the scenario after the thresholds are sent. It uses the default VR-Link thresholds, or the values in the *vrfSimSettings.xml* file, if specified.
- If an object was instantiated with a customized VR-Link *DtThresholder*.

➤ **To configure performance settings:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Performance Settings page (Figure 47).

Figure 47. Performance Settings page



3. Adjust the Visual Quality slider to your desired preference for visual quality versus network performance.
4. To have VR-Forces calculate the smoothing period, select Override Smoothing Period. To specify the smoothing period yourself, select Use Specified Smoothing Value and use the slider to set the smoothing period you want to use.

6.9.1 Enabling Configuration of Performance Settings

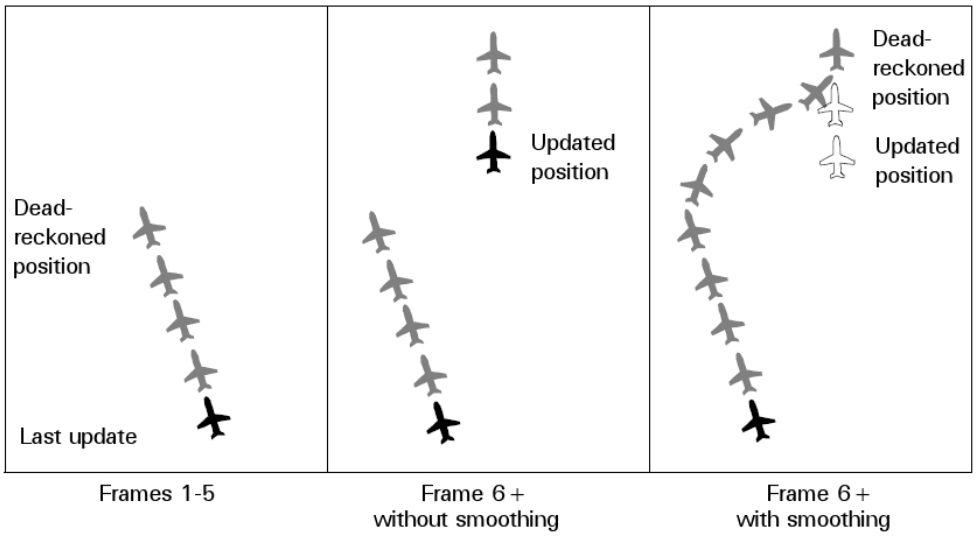
The **dr-allow-gui-overrides** configuration parameter, specified in each OPE file determines whether a given entity type can have its translation and rotation thresholds overridden by the value set by the Performance Settings page.

6.10. Trajectory Smoothing

From the time VR-Forces receives a state update message for a simulation object until it receives the next update, it calculates the location of the simulation object based on the heading and acceleration in the most recent state update. This process is called dead-reckoning. When the next state update arrives for the simulation object, VR-Forces moves it to the new position. If the actual position is significantly different from the dead-reckoned position, the simulation object could appear to jump from the dead-reckoned position to the actual position. VR-Forces can eliminate these discontinuities by smoothing trajectories and rotation paths.

Figure 48 illustrates dead-reckoning and smoothing.

Figure 48. Dead-reckoning and smoothing



The smooth period is the period of time over which smoothing takes place. The default period is 1/10 second. If course corrections appear abrupt with this setting, you may want to increase the smooth period.

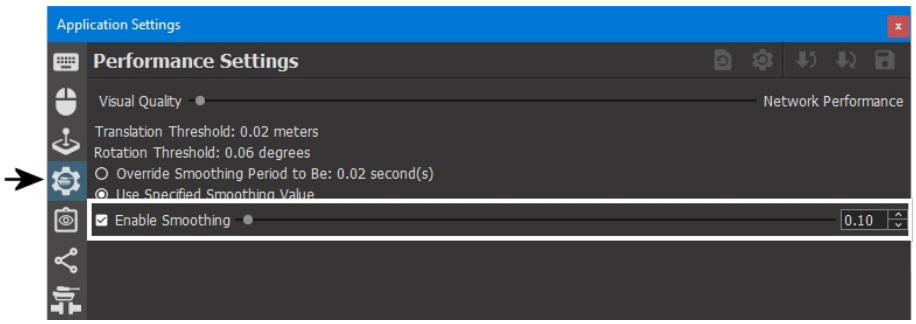
> Smoothing makes fast-moving entities, such as fixed wing aircraft, look better. Slow moving entities, such as dismounted infantry, may look worse. If your simulation is mostly one or the other of these types of entities, enable or disable smoothing as appropriate for the best visualization experience.

6.10.1 Configuring Trajectory Smoothing

➤ **To configure trajectory smoothing:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Performance Options page (Figure 49).

Figure 49. Smoothing settings



3. Select the Use Specified Smoothing Value option. The Enable Smoothing check box becomes enabled.

4. To enable or disable smoothing, select or clear the Enable Smoothing check box.
5. To change the smooth period, adjust the Smoothing Period slider or type a value, in seconds, in the box.

6.10.2 Configuring Articulated Part Smoothing

Articulated parts are also dead-reckoned if velocities are provided for their values. This dead-reckoning can be smoothed in a manner similar to trajectory smoothing.

➤ **To configure articulated part smoothing:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 499 on page 1249).
3. To enable or disable articulated part smoothing, select or clear the Enable Articulated Part Smoothing check box.
4. To change the articulated part smooth period, adjust the articulated part Smoothing Period slider, or enter a value in seconds.

6.11. Tuning the Target Frame Rate

The `--targetFrameRate` command line option and *vrfSim.mtl* parameter allow you to specify the frame rate above which the VR-Forces sim engine should begin to sleep in order to give up CPU time to other processes. This parameter is only applicable to variable frame mode. The fixed frame modes already have a built in sleep that gives up unused time. Increasing the target frame rate prevents the sim engine from sleeping only if it cannot achieve that rate—ultimately, it is up to the operating system to determine how much CPU time each process receives.

6.12. Coloring Draw Calls

One of the factors that affects performance is the number of draw calls it takes to render a scene. VR-Forces lets you color the draw calls in a scene so that you can see which objects could be creating performance problems.

When you color draw calls, all the drawables in the scene are randomly colorized. The colors themselves do not convey any meaning. The important information is the number of draw calls required for a model. For example, if a vehicle has 25 different colored parts, you know that it requires 25 draw calls and you may want to edit the model to use fewer of them.

Figure 50 illustrates a terrain with the feature tiles colorized.

ⓘ Indirect geometry groups draw calls together to ensure that there are fewer draw calls. Indirect rendering for entities is enabled by default. To learn more, see "Indirect Rendering of Terrain" or "Indirect Rendering for Entities" in the *Adding Content to MAK ONE Applications Reference Manual*.

Figure 50. Colorized drawables on terrain



The colors do not update as new information comes into the scene. Therefore, you can refresh the colorization to see your changes by changing the Component Visualization setting to a different visual debug mode and then back to Debug Texture.

➤ **To visualize draw calls:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40 on page 203).
3. In the Debug Visuals list, choose Colorize Drawables.

The colors may looked washed out if high dynamic range is enabled. See "52.9. Using High Dynamic Range Lighting" on page 1097.

6.13. Configuring VSync

VSync (vertical synchronization) synchronizes VR-Forces's frame rate with your monitor's refresh rate. This prevents a visual anomaly called tearing, in which parts of two frames of data can be visible at the same time, but it caps the maximum render frame rate based on the display. Typically this cap is 60 Hz.

In some cases, if the frame rate cannot quite keep up with the refresh rate, VSync can force the frame rate to a noticeably lower rate. To avoid this problem, we recommend that you enable triple buffering for your video card. You must do this in the video card's configuration application; VR-Forces cannot do this for you.

By default, VSync is disabled. If you want to enable it, start up with the `--enableVSync` command-line option, or you can configure your video card to force it to be on.

For `--enableVSync` to work, your graphics card must be set to "Use application setting".

6.13.1 Managing 3D Settings and Multiscreen Performance

In a configuration with remote displays on different screens using the same host computer, you can encounter frame rate issues when vertical sync (VSync) is enabled.

If you see a very low frame rate or a frame rate that cycles between high and very low, first make sure that all monitors are using the same refresh rate (usually 60 Hz).

➤ **To check the refresh rate on a monitor:**

1. Open the NVIDIA Control Panel.
2. Choose the Change Resolution task.
3. Check the Refresh Rate.
4. If you made any changes, click Apply.

If that does not solve the problem, try changing the Vertical Sync setting in the NVIDIA Control Panel.

➤ **To edit the Vertical Sync settings:**

1. Open the NVIDIA Control Panel.
2. Choose the Manage 3D Settings task.
3. In the Global Settings tab, check the Vertical Sync setting. If Vertical Sync is set to On, try setting it to Use the 3D application setting or Off.
4. In the Program Settings tab, check whether vrvIG or vrvStealth are included in the list. If either is listed, try setting Vertical Sync to Use the 3D application setting or Off.
5. Click Apply.

6.14. Adaptive Tick Rate

The Adaptive Tick Rate check box in the Entity Display Settings controls whether the tick rate for entities adjusts based on their distance from the observer. You can enable adaptive tick rate to allow VR-Forces to support a large number of entities that need to be updated in a scenario. When adaptive tick rate is enabled, entities update their state (for example, position and orientation) less frequently when they are further from the observer.

Please note that there can be display issues with high speed objects, such as aircraft, when adaptive tick rate is enabled. It is intended for use when there are many human entities.

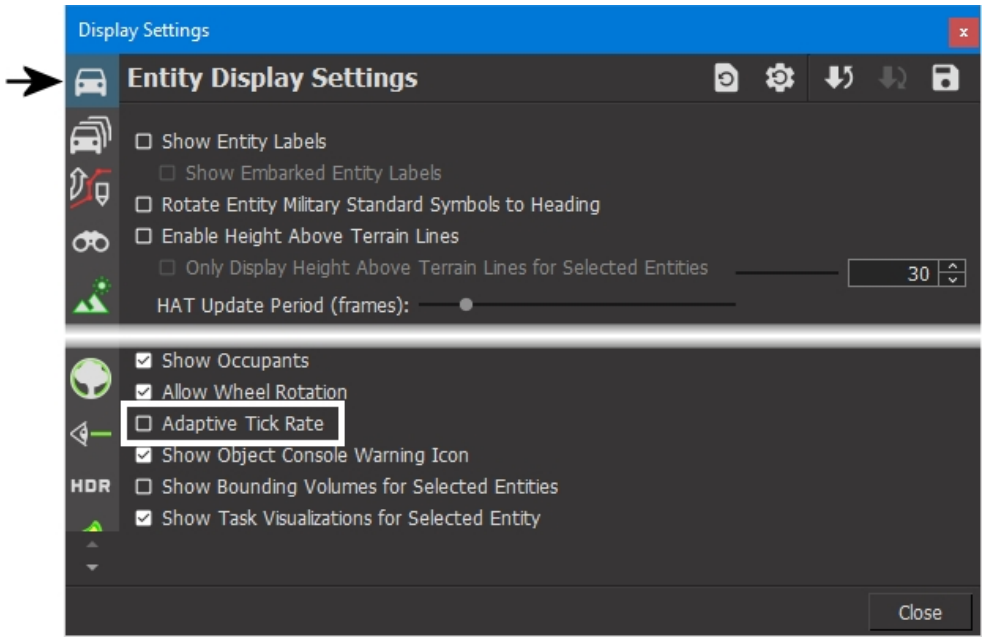


The distance used for adaptive tick rate has an inner radius and an outer radius. If an entity is farther than 1000 meters from the observer (inner radius), the updates are about once for every six frames. The update then decreases until 10,000 meters (outer radius), where the updates occur once every 30 frames. These distances are the default values.

➤ **To enable or disable adaptive tick rate:**

1. Choose **Settings > Display**.
2. Select the Entity Display Settings page.
3. Select or clear the Adaptive Tick Rate checkbox.

Figure 51. Adaptive Tick Rate in Entity Display Settings



6.15. Preloading Models

When VR-Forces discovers simulation objects on the network, it must load the visual models for these objects. This can affect performance. It is particularly noticeable when VR-Forces first connects to an exercise, but can occur at other times if there is a sudden influx of new objects. You can reduce the likelihood of this problem by loading models before you connect to an exercise.

To preload models, you create a list of the models that you expect to need. VR-Forces loads these models when it starts up. You can also load them immediately.

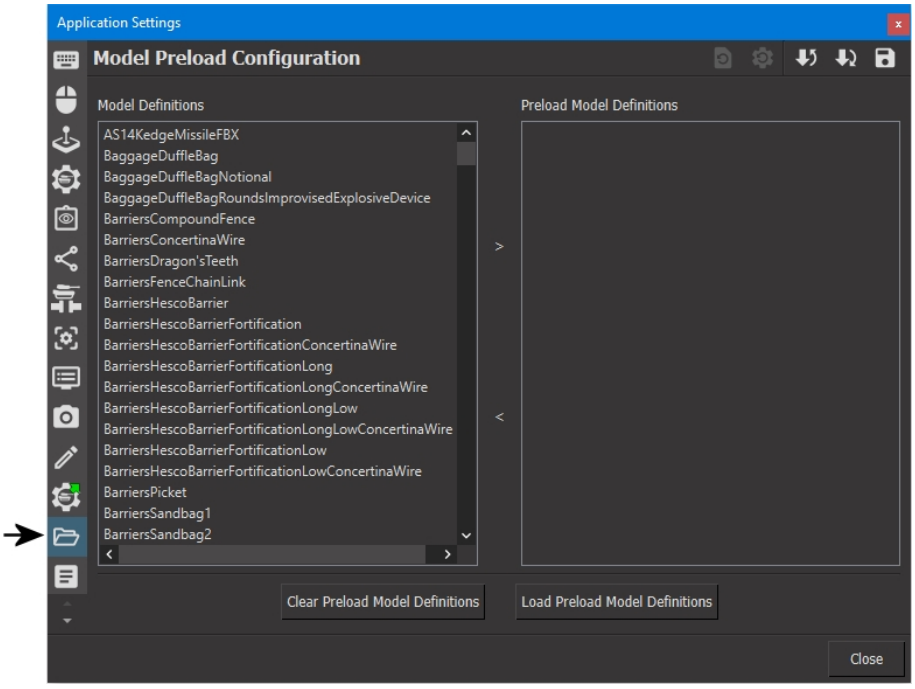
While VR-Forces is participating in an exercise, you can capture a list of models that have not been preloaded. You can then add this list of models to the default preload list or specify the list in a command-line argument.

By default, the list of models to be loaded is saved to *./appData/settings/vrfGui/default_ModelPreload.mpsx*. You can export the list to a different file name. You can also import and merge lists.

➤ **To configure the list of models for preloading:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Model Preload Configuration page (Figure 52).

Figure 52. Model Preload Configuration page



3. In the Model Definitions list, select the models that you want to preload. Use the standard methods of multiple object selection.
4. Click the right-facing arrow to move the models into the Preload Model Definitions list.
5. To load the models immediately, click Load Preload Model Definitions.

6.15.1 Command-Line Options for Preloading Models

By default, when VR-Forces starts, it loads the models listed in *default_ModelPreload.mpsx*. You can specify that it load a different model list with the `--modelPreloadFileName` command-line option. The syntax is:

`--modelPreloadFileName` *`filename`*

where *`filename`* is the name of the file to load.

While VR-Forces is participating in an exercise, it can capture a list of models that were not preloaded. You can then merge this list with the default preload file or specify it as an alternative to the default file on the command line. To capture a list of models, use the command:

`--discoveredModelPreloadFileName` *`filename`*

where *`filename`* is the name of the file to save the list to.

6.16. DI-Guy Performance Settings

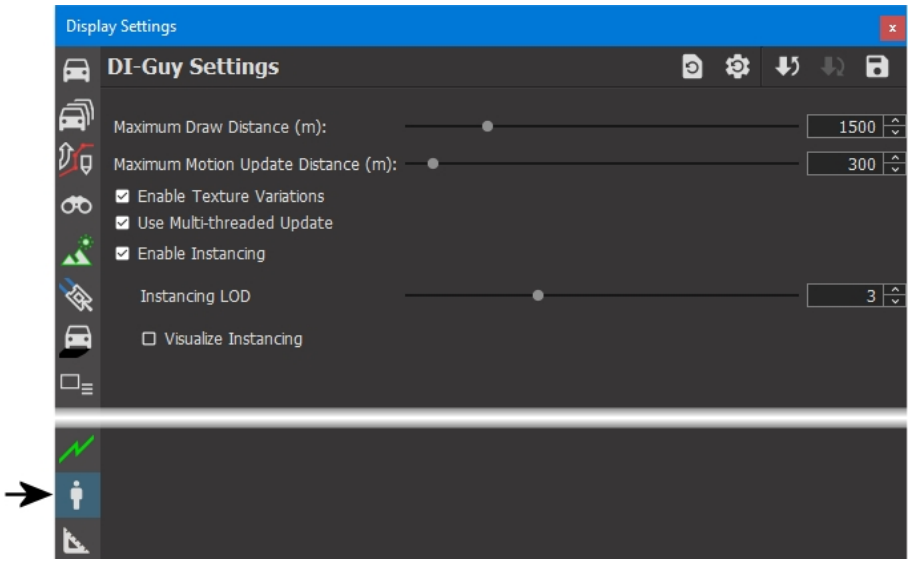
These settings affect performance of DI-Guy human characters:

- Maximum draw distance. VR-Forces will not render characters that are farther away from the observer than this distance.
- Maximum motion update distance. The distance at which VR-Forces stops animating DI-Guys unless they are changing posture.
- Use multi-threaded update. Developers who are debugging DI-Guy may want to disable multi-threaded update to make it easier to debug.
- Enable instancing. Enabling instancing improves performance by reusing model data for similar characters. You can set the LOD beyond which instancing is used. You can visualize instanced characters.

➤ **To specify DI-Guy performance settings:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the DI-Guy Settings page (Figure 53).

Figure 53. DI-Guy Settings page



3. To set the maximum draw distance, adjust the Max Draw Distance slider or change the value in the box.
4. To change the distance at which DI-Guys stop animating, adjust the Maximum Motion Update Distance slider.
5. Optionally, select or clear the Use Multi-threaded Update check box.
6. Optionally, select or clear the Enable Instancing check box.
7. If instancing is enabled, adjust the Instancing LOD slider or change the value in the box.
8. Optionally, select or clear the Visualize Instancing check box.

6.17. Specifying Terrain Categories

To optimize terrain intersections that VR-Forces performs for different purposes and for different entity types, you can adjust the terrain categories on a per-platform (OPE file) basis.

Each platform file specifies **los-terrain-category-mask** and **movement-terrain-category-mask**. The value for these parameters can be specified to be any combination of:

- TerrainSkin
- Structure
- Vegetation

Any category of terrain not included in the mask will potentially be ignored for that type of intersection by entities using the platform file in question. Note that some terrain data sources do not specify a category, which means that they are always included because they have no category that VR-Forces can recognize.

You should specify the minimum categories for each platform type that are needed for your scenario and expected behavior. For example, the default VR-Forces entity-level SMS does not include vegetation for the **movement-terrain-category-mask** for humans because it is not expected that humans will walk on trees. This allows VR-Forces to avoid paging in vegetation data near humans as they move.

Similarly, the default configuration for fixed-wing aircraft platforms includes only TerrainSkin for both line-of-sight (LOS) and movement masks. Because these are typically fast-moving entities with long range sensors, terrain intersection checks and terrain paging can be quite costly for them. The entity-level SMS that ships with VR-Forces assumes that these entities do not need to land or taxi on buildings or trees, therefore these are not in the movement mask. VR-Forces also assumes that buildings and trees will not be primary blocks for LOS tests performed by fixed-wing aircraft entities. While this choice makes the terrain paging much faster, there may be scenarios where buildings do need to block the LOS for their sensors and therefore it should be added.

For more information, please refer to the *VR-Forces Developers Guide*.

6.18. Using the Developer Debug Overlay

VR-Forces uses the ImGui API to create an overlay that developers can use to test changes to some of its visual features. VR-Forces includes an overlay for testing ocean settings, shadow settings, and other visual features. It also includes an ImGui test user interface that demonstrates some of its features.

Figure 54. Developer debug overlay



This feature is provided "as is" for the use of interested developers. For complete information about how to use ImGui, please see documentation at https://github.com/ocornut/imgui and other internet resources.

➤ **To enable the developer debug overlay:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 40 on page 203).
3. Select the Developer Debug Overlay check box. The Developer Debug Overlay opens. It has menus and options for Ocean Settings, SpeedTree Settings, osgEarth Network Monitor, and so on.

   You can also open the developer debug overlay by pressing **F4**.

4. To view shadow settings:
   a. In the Display Settings dialog box, select the Shadow Settings page.
   b. Select the Enable Shadow check box.

6.19. Using the MAK Remote Debugger

The MAK Remote Debugger is another tool developers can use to understand what is happening in the VR-Forces simulation engine. This includes sim engine performance, publishing information for entities in the scenario, and detailed information about the terrain.

You can open the MAK Remote Debugger by running *makRemoteDebugger.exe* from the *./bin64* directory. On Windows, you can also choose **MAK Technologies > VR-Forces 5.2 Tools** from the Start menu and then use the MAK Remote Debugger shortcut. When you first start it, it displays information on how to use the debugger. Enter the desired commands in the vrfSim console window to configure and connect the debugger to your VR-Forces instance and view the sim engine details.



II. Scenarios

Whenever you use VR-Forces, you work in the context of a scenario. A scenario consists of everything that needs to be specified to run a VR-Forces simulation. Each scenario has a scenario file (*.scn*) and these supporting files:

- A scenario objects file (*.oob*).
- A plan file (*.pln*).
- An object map file (*.omp*).
- A terrain database for the GUI and sim engine.
- An overlay file (*.ovl*).
- A scripts file (*.spt*). This file contains the scenario scripts, if any.
- A selection groups file (*.sgr*).
- A scenario "extras" file (*.xtr*). This file records hostility relationships and stores spawn pattern templates.
- A saved views file (*.osrx*). If a scenario has captured observer views, they are saved automatically. You can also specify a startup view.
- A settings file that records scenario-specific GUI settings (*.gui_settings*).

By default, scenarios are saved in a compressed zip archive with the extension *.scnx*. If you examine a compressed scenario file in an unzipping application, you will see that it contains the individual files described in the previous paragraphs.

A terrain database does not have to be stored in the same directory as a scenario. It is referenced by the scenario. This lets you use a terrain database, which can be quite large, in more than one scenario, without needing to make multiple copies of it. For more information about terrain databases, see "Section XII. Terrain" on page 1237.

For details about scenario files, see "Chapter 12. Scenario Files" on page 349.



7. Creating and Running Scenarios

This chapter explains how to create, load, run, and save scenarios.

7.1. Creating a Scenario 228
7.1.1 Specifying Multiple Simulation Model Sets 233
7.1.2 Creating Simulation Model Set Configurations 235
7.1.3 Setting the Scenario Starting Date and Time 236
7.2. Importing Objects into Scenarios 238
7.2.1 Importing (Merging) Scenarios 239
7.2.2 Importing and Exporting MSDL 241
7.2.3 Importing Scenario Objects from CSV Files 243
7.2.4 Importing Airspace Control Orders 245
7.2.5 Importing Two-Line Element Sets 246
7.2.6 Loading Object Tags 246
7.3. Saving a Scenario 247
7.3.1 Saving a Previously Saved Scenario 248
7.3.2 Saving a Scenario to a New Name or a Different Format 249
7.4. Loading a Scenario 249
7.4.1 Loading a Recently Loaded Scenario 251
7.4.2 Loading a Scenario from the Command Line 251
7.4.3 Editing Scenario Properties 251
7.4.4 Editing the Scenario Description 253
7.4.5 Navigating Scenario Directories 254
7.4.6 Sample Scenarios 254
7.5. Load Balancing Scenarios 255
7.5.1 Load Balancing an Existing Scenario on Load 255
7.5.2 Load Balancing After Loading the Scenario 257
7.5.3 Loading a Load-Balanced Scenario 258
7.6. Running a Scenario 260
7.6.1 Changing the Simulation Speed 260

7.6.2 Pausing a Scenario 262
7.6.3 Rewinding a Scenario 262
7.6.4 Closing a Scenario 263
7.7. Reinitializing a Scenario 264
7.8. Checkpoints and Snapshots 264
7.8.1 Saving Checkpoints 265
7.8.2 Deleting Checkpoints 267
7.8.3 Creating Snapshots 267
7.8.4 Clearing the Snapshot List 268
7.9. Pausing a Scenario Automatically 268
7.9.1 Setting the Scenario End Time for a New Scenario 268
7.9.2 Setting the Scenario End Time for an Open Scenario 269
7.10. VR-Forces and Batch Mode 269
7.10.1 The Batch File 269
7.10.2 Creating a Batch File 270
7.10.3 Editing a Batch File 270
7.10.4 Running VR-Forces in Batch Mode 272
7.10.5 Recording Batch Scenarios 273
7.11. Sending Run and Pause Messages to Simulation Participants 273
7.12. Recording VR-Forces Simulations with the MAK Data Logger 274

7.1. Creating a Scenario

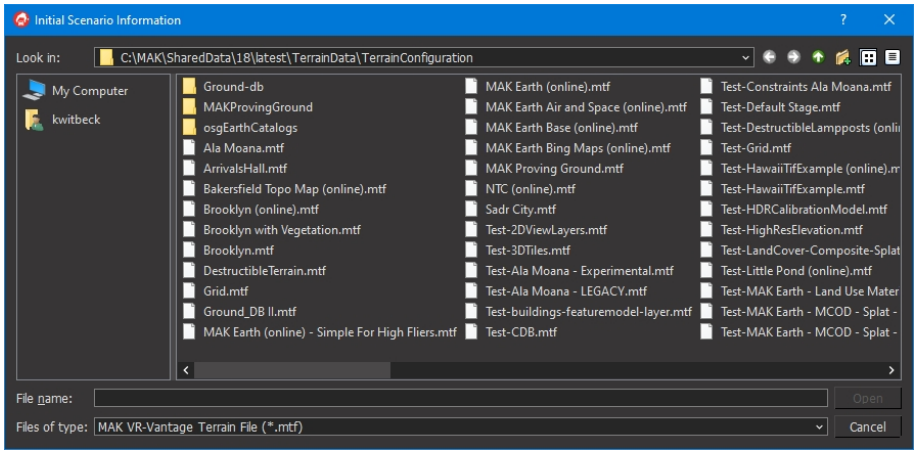
To use VR-Forces to simulate simulation objects, you must load a scenario or create a new one. When you start VR-Forces, the Scenario Startup screen asks you whether you want to create or load a scenario (unless you have disabled this feature). Thereafter, to create a scenario, use the procedure in this section. (For details about creating a scenario from the Scenario Startup screen, see "4.1.1 Starting VR-Forces from the VR-Forces Launcher" on page 129.)

> If you are using multiple GUIs in the same session, when one of them creates a scenario, the terrains in all other GUIs are closed (unless the terrain is the same as that for the new scenario) and the terrain for the new scenario is loaded.

➤ **To create a scenario:**

1. Choose **File > New Scenario**, or click the New Scenario button ( ) on the File Toolbar. The Initial Scenario Information dialog box opens (Figure 55). By default, it opens in *./userData/terrains*, the directory that contains the terrain files shipped with VR-Forces. (For a brief description of the databases supplied with VR-Forces, see "Terrain Databases Provided with VR-Forces" on page 1238. For details about terrain files, see "Section XII. Terrain" on page 1237.)

Figure 55. Initial Scenario Information dialog box

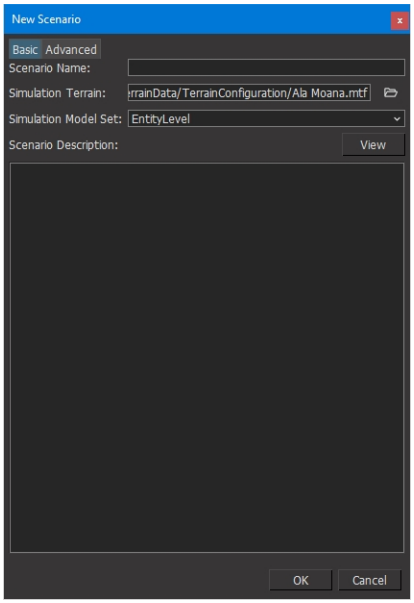


2. In the file list, select a terrain database. (MTF is the preferred file type.) A database must be a flat earth, geocentric, UTM, or Lambert Conformal Conic database. By default, this terrain is used for both the GUI and sim engine.

   If you choose an MTF file and an MTD file with the same name exists, the sim engine loads the MTD file. (MTD is a legacy format that may be present for older terrains.)

3. Click Open. The New Scenario dialog box opens (Figure 56).

Figure 56. New Scenario dialog box



4. Select the simulation model set (SMS) for the type of scenario you want to run: entity-level or aggregate-level.

5. Optionally, set other basic scenario parameters. Table 13 describes the basic parameters.

Table 13. Basic scenario parameters

Parameter	Description
Scenario Name	Specifies the name of the scenario. This is the name used to identify the scenario when you join a session. If you do not specify a name, the scenario file name is used.
Simulation Terrain	Specifies the terrain to be used by the sim engine. By default, this is the terrain you selected in the Initial Scenario Information dialog box. However, you can change it if you want to.
Simulation Model Set	The simulation model set to use. For entity-level scenarios, choose *EntityLevel.sms* or an SMS similar to it. For aggregate-level scenarios, choose *AggregateLevel.sms* or a similar SMS.
Scenario Description	The scenario description lets you provide a description of the scenario. This may be useful to remind you or other VR-Forces users of its purpose or special details. You can enter the description as plain text or as HTML. If you use HTML, when the description is displayed in the Scenario Properties dialog box, it is displayed using the HTML formatting.
Attach Components to Remote Entities On	Specifies whether to attach components to remote entities, and if so, the sim engines on which to manage the components that are being attached to remote entities. Unlike other scenario parameters, you can change this value when you load a scenario. For details about configuring components for attachment to remote entities, see "82.1. Attaching Components to Remote Entities" on page 1592.

For more information about these parameters, see "12.2.1 Scenario Parameters" on page 351.

6. Optionally, select the Advanced tab (Figure 57) and set advanced scenario parameters (overriding the default settings).

Figure 57. New Scenario dialog box, Advanced tab

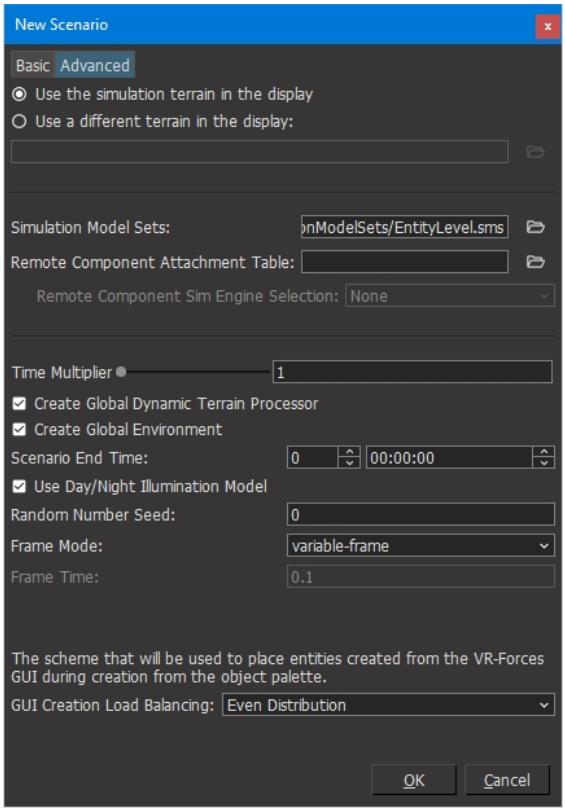


Table 14 describes the parameters you can set.

Table 14. Advanced scenario parameters

Parameter	Description
Use the Simulation Terrain in the Display	If selected (the default), the scenario uses the same terrain for the GUI and sim engine.
Use a Different Terrain for the Display	If selected, lets you specify the terrain to be used by the GUI.
Simulation Model Sets	Lets you specify simulation model sets other than or in addition to the default SMSs listed on the Basic tab. A scenario must specify at least one simulation model set. For more information, see "7.1.1 Specifying Multiple Simulation Model Sets" on page 233. Default: *EntityLevel.sms*.

Table 14. Advanced scenario parameters (continued)

Parameter	Description
Component Attachment Table	If you plan to attach components to remote entities, specify the component attachment table that you want to use. For details about configuring components for attachment to remote entities, see "82.1. Attaching Components to Remote Entities" on page 1592.
Time Multiplier	Specifies a time multiplier for the scenario. For HLA time-managed federations, this parameter is ignored. Default: 1.0 (real-time).
Create Global Dynamic Terrain Processor	If selected, the global dynamic terrain processor monitors the entire terrain for dynamic terrain changes. If disabled, dynamic terrain is not supported unless you create a dynamic terrain area.
Create Global Environment	If selected, global weather is supported. If cleared, the settings on the Environment Conditions dialog box, Weather page have no effect on the scenario. (You can still create weather zones.)

The Create Global Dynamic Terrain Processor and Environment options are available only when creating new scenarios. If you want to remove the object from an existing scenario, load the scenario and then use the Sim Engine Load Distribution dialog box to delete it. Choose **Simulation > Load Distribution**, then right-click on the object in the list and choose **Delete**.

Scenario End Time	The simulation time at which to automatically pause the scenario. For more information, see "7.9.1 Setting the Scenario End Time for a New Scenario" on page 268.
Use Day/Night Illumination Model	If selected, use an ephemeris model in the sim engine to calculate illumination. This affects sensors. If cleared, the entire world is fully illuminated regardless of the time of day. For more information, see "11.3.3 Choosing the Illumination Model" on page 332.
Random Number Seed	Specify a random number seed as an integer >= 0.
Frame Mode	Select a frame mode from the list.

Table 14. Advanced scenario parameters (continued)

Parameter	Description
Frame Time	Specifies the length of a frame, in seconds. This parameter is not enabled if Frame Mode is variable-frame.
GUI Creation Load Balancing	Specifies how to load balance simulation objects created in the GUI when you have more than one simulation engine. If you select Even Distribution, VR-Forces automatically distributes objects across the available sim engines. If you select Rule Based Distribution, the Rule Set list becomes available and you can choose an existing rule set or create a new one. See "Rule-Based Load Balancing" on page 256.

For more information about these parameters, see:

- "3.4.1 Simulation Time" on page 120.
- "3.4.3 Exercise Clock Modes" on page 122.
- "12.2.1 Scenario Parameters" on page 351.

7. Click OK. The scenario information is displayed and the terrain is displayed in the Plan View (2D) observer mode.
8. Add the simulation objects and tactical graphics that you need for your simulation. For information about adding simulation objects and tactical graphics, see "Chapter 14. Creating and Placing Objects" on page 371.
9. Save the scenario. For details about saving scenarios, see "7.3. Saving a Scenario" on page 247.

7.1.1 Specifying Multiple Simulation Model Sets

You can specify more than one simulation model set for a scenario. One strategy for extending VR-Forces is to use an SMS provided with VR-Forces for the majority of objects and to create a separate SMS for customized simulation objects. As you upgrade to newer versions of VR-Forces, you can more easily migrate your customized SMS to the new version because your changes have not been integrated into the default SMS. (You can also include the SMSs provided by VR-Forces in your custom SMS. In that case you would just load the custom SMS and not multiple SMSs. For more information, see "68.3. Simulation Model Sets" on page 1309.)

If you routinely use more than one SMS, consider creating an SMS configuration. For details, see "7.1.2 Creating Simulation Model Set Configurations" on page 235.

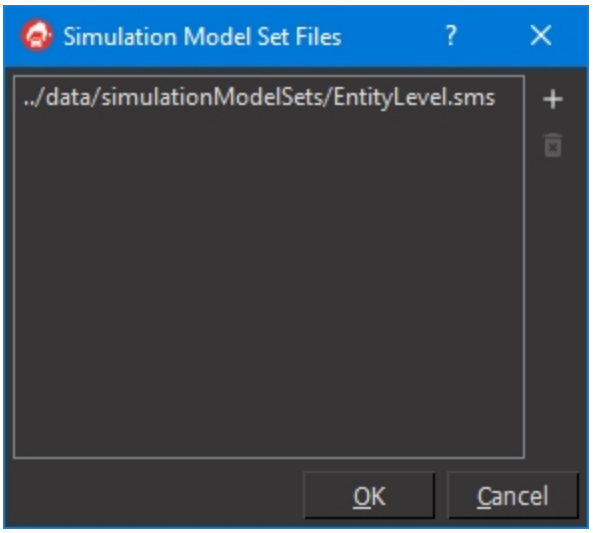
When you load multiple SMSs, if there is any overlap in the objects they specify, the last SMS loaded overrides previous SMSs.

> ⚠ Do not use an entity-level SMS and an aggregate-level SMS together. The simulation objects from the two SMSs will not be able to interact.

➤ **To specify multiple simulation model sets for a scenario:**

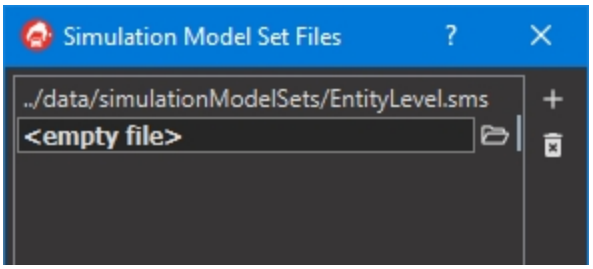
1. In the New Scenario dialog box, click the Browse button on the Simulation Model Set line. The Simulation Model Set Files dialog box opens. It lists the SMS files to be loaded for this scenario.

Figure 58. Simulation Model Set Files dialog box



2. In the Simulation Model Set Files dialog box, click Add Model Set. A blank line is added to the window (Figure 59).

Figure 59. Add simulation model set



3. Click the Browse button next to the blank line. The Choose File dialog box opens.
4. Select a simulation model set and click Open. The SMS is added to the text box.
5. Click OK.
6. Complete the New Scenario dialog box.

7.1.2 Creating Simulation Model Set Configurations

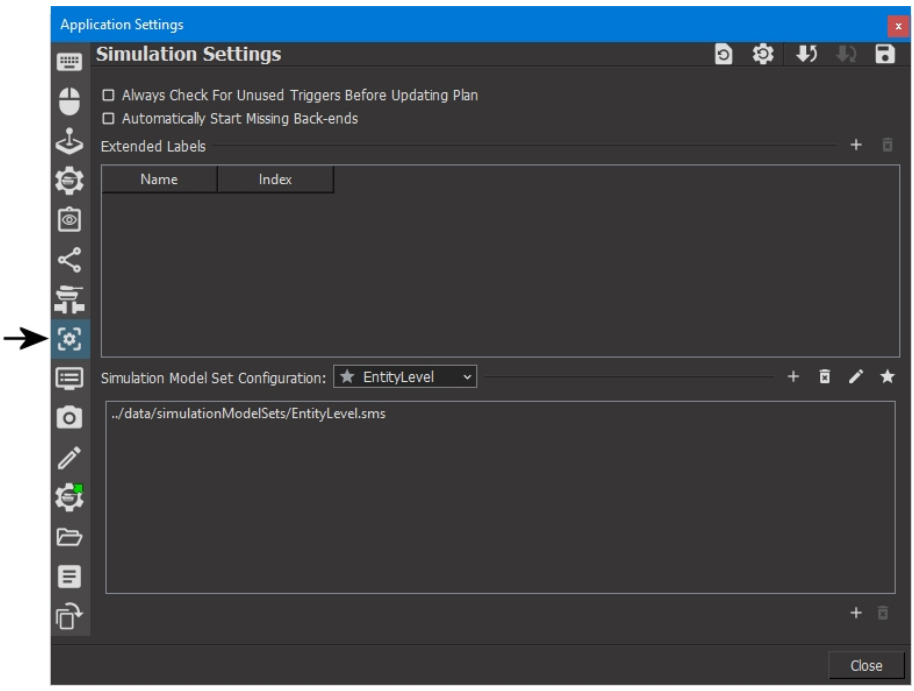
When you create a new scenario, you specify an SMS in the Initial Scenario Information dialog box (see "7.1. Creating a Scenario" on page 228 for details). The Simulation Model Set list lists all SMSs in *./data/simulationModelSets*. The SMSs are listed in alphabetical order. If you want a particular SMS to be the default SMS you can create an SMS configuration and specify that it be the default. You can create as many SMS configurations as you want. An SMS configuration can have more than one SMS in it. So if you routinely create scenarios that use multiple SMSs, creating a configuration can save you from the necessity of always specifying the additional SMSs on the Advanced tab of the New Scenario dialog box (as described in "7.1.1 Specifying Multiple Simulation Model Sets" on page 233).

The Simulation Model Set list in the New Scenario dialog box lists all SMS configurations and all SMSs. It does not duplicate names, so if an SMS configuration has the same name as an SMS, only the SMS configuration is listed.

➤ **To create a simulation model set configuration:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 60).

Figure 60. Simulation Settings page



3. Click the Add button (+) on the Simulation Model Set Configuration line. The Configuration dialog box opens.
4. Type a name for the configuration.

5. Click OK. The new configuration is added to the list.
6. Click Add Model Set. A line is added to the window.
7. Click the Browse button at the end of the new line. A file chooser dialog box opens.
8. Select the SMS you want to put in this configuration.
9. Click Open. The SMS is added to the list.
10. Optionally, add additional SMSs to this configuration.
11. Click Close.

Specifying the Default SMS Configuration

You can specify that a particular SMS configuration be selected in the Initial Scenario Information dialog box whenever you create a new scenario.

> Specifying an SMS as the default SMS automatically removes that designation from any other SMS.

➤ **To specify the default SMS configuration:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 60 on the previous page).
3. In the Simulation Model Set Configuration list, select the configuration that you want to be the default.
4. Click the Star button (★). A star displays next to the configuration name in the list, indicating that it is the default configuration.

7.1.3 Setting the Scenario Starting Date and Time

By default, when you create a scenario, the starting simulation date and time is the current system date at 10:00 A.M. local time, where the local time zone is determined by the longitude of the first simulation object that you add to the scenario.

You can change the date and time at which a scenario starts in either of the following ways:

- A date and time other than the default, but still at the location where the first simulation object is created.
- Use a date, time, and time zone without regard to where the first simulation object is created.

By default, before you create the first simulation object, as you move the observer through the terrain, the time shown on the Time of Day toolbar and in the Date and Time page changes to show you the time at the location of the observer relative to the selected time zone. (The Time of Day toolbar and the Date and Time page defaults to Coordinated Universal Time (UTC, also called Zulu time).) Furthermore, the sun moves with the observer so that the observer location is always in daylight.

The time zone is an approximation. It does not take into account unusual time zone boundaries or daylight saving time. The time zone is set once. If you create a simulation object and then delete it, the location of that entity still determines the time zone. Furthermore, if for some reason you were to create a scenario and then run it without creating any simulation objects, if at any point you created a simulation object, the starting date and time would be reset to the default value based on the location of that entity.

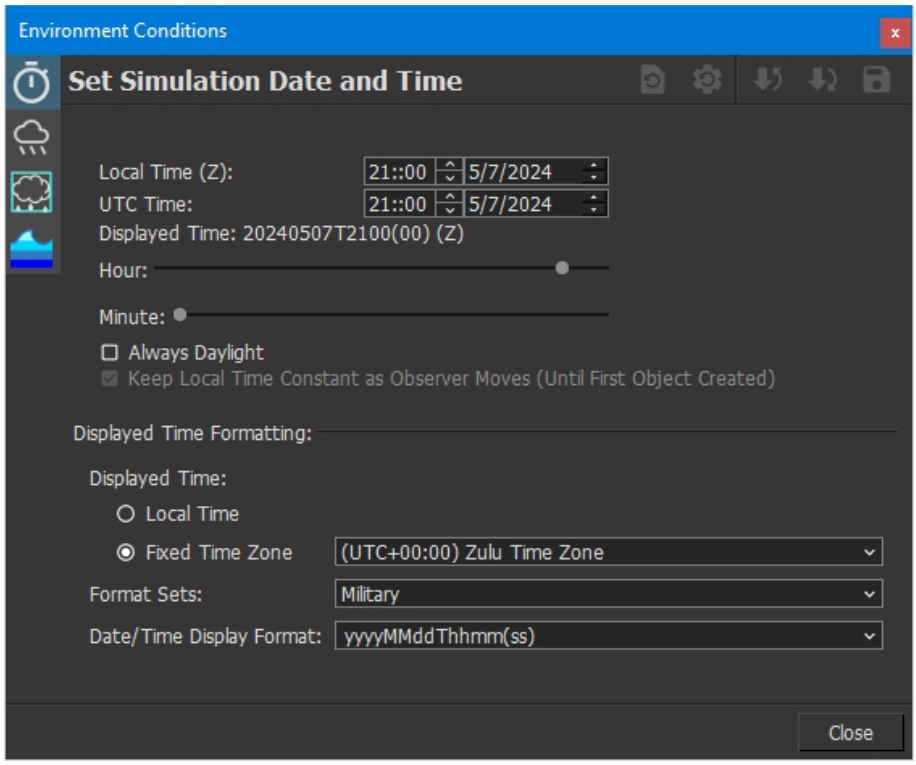
You can change the starting date and time when you create a scenario.

For more information about simulation time, see "11.3. Setting the Date and Time of Day" on page 330.

➤ **To set the scenario start date and time using the time zone of the first simulation object:**

1. Begin the scenario creation process, as described in "7.1. Creating a Scenario" on page 228.
2. Before you create the first simulation object, choose **Simulation > Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
3. Select the Date and Time page (Figure 61).

Figure 61. Set Simulation Date and Time page



4. Change the date and current time to the desired date and current time.
5. Create a simulation object in the desired location.

➤ **To set the scenario start time and time zone without respect to the first simulation object:**

1. Begin the scenario creation process, as described in "7.1. Creating a Scenario" on page 228.
2. Before you create the first simulation object, choose **Simulation > Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
3. Select the Date and Time page (Figure 61).
4. Clear the Keep Local Time Constant as Observer Moves (Until First Object Created).
5. Set the date, time, and time zone that you want the scenario to start at.
6. Optionally, specify the date and time format (standard or military) and the display format.
7. Continue the scenario creation process.

7.2. Importing Objects into Scenarios

If you frequently need to create scenarios with the same simulation objects, scenario-level scripts, or other scenario components, consider using one or more of these strategies:

- Create simulation object groups. Simulation objects groups are heterogeneous groups of simulation objects and tactical graphics that you can add to any scenario. Like the individual simulation objects that you can add to a scenario, they are not tied to a terrain location until you add them. Use them if you want to create a repeatable set of objects that do not fit into a unit hierarchy. For more information, see "14.14. Simulation Object Groups" on page 400.
- Use scenario import. If you have a base scenario that you want to use repeatedly, create the scenario, then open it and rename it to preserve the original, or open a new scenario and import the base scenario. You can do this with a fully realized scenario or a scenario layout that you want to add to other scenarios. For example, you might want to create a town with background traffic for repeated use with different training problems added to it. Since this is a scenario, it is tied to the terrain location you use when you create it. Alternatively, you might have a set of scripts that you do not want to promote to system scripts. You can save them in a scenario and import the scenario into the scenarios for which you want to use the tasks. For more information, see "7.2.1 Importing (Merging) Scenarios" on the facing page.
- Import MSDL. If you have a MSDL entity laydown, you can import that into different scenarios as needed. For more information, see "7.2.2 Importing and Exporting MSDL" on page 241.
- Import point, linear, or areal objects from a CSV file. For details, see "7.2.3 Importing Scenario Objects from CSV Files" on page 243.
- Export an Order of Battle and import it into other scenarios. For details, see "Chapter 22. Creating and Using an Order of Battle" on page 489.
- Export tactical graphics to shapefiles. For details, see "46.3. Exporting Tactical Graphics as Shapefiles" on page 1008.

7.2.1 Importing (Merging) Scenarios

VR-Forces users often assign scenario development to multiple people or teams. Then they merge the separately-developed scenarios into a master scenario. You can easily merge scenarios by importing them. When you import a scenario, you can choose to rename simulation objects to avoid duplicate names, to import scenario events, and to create a selection group for the imported simulation objects.

Some object names in the scenario you are importing might conflict with (that is, might be the same as) object names in the base scenario. You have three options for handling object name conflicts:

- Allow duplicate names in the merged scenario.
- Give every object in the imported scenario a new name by adding the prefix specified in the Deconflicting Name field. The Deconflicting Name prefix is applied to every object name, not just conflicting names, except as noted below in "Object Names in Merged Scenarios" on page 241.
- Change only conflicting names in the imported scenario.

The procedure in "Importing a VR-Forces Scenario" on the next page explains how to use each of these options.

> ⚠ If you import a scenario that was saved (checkpointed) with a simulation time that is different from the current time of the scenario into which you are importing it, there may be unexpected behaviors due to timing issues. For example, an entity executing a Wait Duration task may wait for a longer or shorter time than you expect.

What Is (and Is Not) Included in the Import

Not everything associated with a scenario is included.

What is imported:

- Simulation objects without conflicting UUIDs, including entities, units, and tactical graphics.
- Simulation objects with conflicting UUIDs, only when Discard Duplicate Simulation Objects check box is not selected.
- Scenario events, only when Import Scenario Events is selected.
- Global plans.
- Synchronization matrices.

What is not imported:

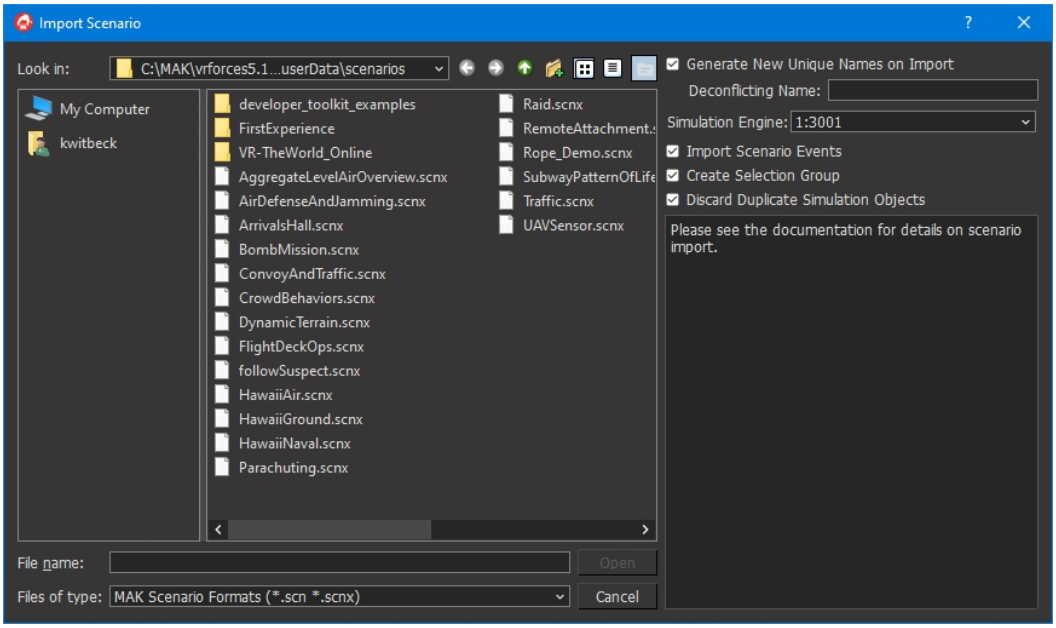
- The SMS associated with the imported scenario.
- The global environment, including the weather, date, and time of day.
- The terrain.
- Observer views.
- Dynamic terrain changes made by the global dynamic terrain manager object.
- Simulation objects with conflicting UUIDs, only when Discard Duplicate Simulation Objects check box is selected.

Importing a VR-Forces Scenario

➤ **To import a scenario:**

1. Open a scenario.
2. Choose **File > Import VR-Forces Scenario**. The Import Scenario dialog box opens (Figure 62).

Figure 62. Import Scenario dialog box



3. Select the scenario that you want to import.

4. Execute only one of the following:
   - To allow duplicate names in the merged scenario, clear the Generate New Unique Names on Import check box.
   - If you want VR-Forces to give every object in the combined scenario a unique name:
     - Make sure the Generate New Unique Names on Import box is checked.
     - In the Deconflicting Name field, enter a text string for VR-Forces to prepend to object names in the imported scenario. (VR-Forces adds the string to all object names, not just to those that match names in the base scenario, except as noted below in "Object Names in Merged Scenarios" below.)
   - To change only conflicting names in the imported scenario:
     - Make sure the Generate New Unique Names on Import box is checked.
     - Clear the Deconflicting Name field.
5. Optionally, specify whether to import scenario events.
6. Optionally, specify whether to create a selection group.
7. Optionally, specify whether to discard duplicate simulation objects.

Object Names in Merged Scenarios

The deconflicting names feature treats units (platoons, for example) and platforms (vehicles) differently from other objects:

- VR-Forces applies the prefix you enter in the Deconflicting Names field to every unit in the imported scenario, whether its name conflicts or not.
- VR-Forces renames platforms in the imported scenario that have names that conflict with platforms in the base scenario, but does not use the prefix to do it. Instead, VR-Forces changes the number at the end of the platform's name, so a tank called M1A2 1 in the imported scenario might become M1A2 4 in the merged scenario.

In addition, if adding the Deconflicting Names prefix would make an entity's name too long, VR-Forces gives the entity a new unique name that does not include the prefix.

7.2.2 Importing and Exporting MSDL

Military Scenario Definition Language (MSDL) is a SISO standard for describing military scenarios using an XML format. It allows for interchange of scenario descriptions between simulation applications and systems that use different internal formats. You can import MSDL files into a VR-Forces scenario.

VR-Forces does not import all of the information that can be in an MSDL file. It imports and exports:

- Simulation objects and their locations. Locations are always exported in geocentric coordinates. Files for import must have valid locations.

- Force relationships and hostility relationships. Any force with the word "neutral" in it (not case sensitive) is considered a neutral force, and all other forces are considered neutral to it unless specifically marked as hostile in the VR-Forces hostility matrix. Other forces not marked as hostile to one another are considered friendly.
- Country codes. If the scenario's hostility matrix does not have countries assigned to forces, the country code is empty. You should define countries for the forces, including the Neutral force, to have country codes in the MSDL file.
- Unit formations.
- Tactical graphics.
- Area of Interest (bounding box).

If you try to import a unit type that is not available in the Simulation Object Editor, you will not be able to import it because it cannot be assigned to a unit type. You can add empty units in the Simulation Object Editor that will then be available for MSDL import. For more information, see "Chapter 71. Editing Units for Entity-Level Scenarios" on page 1405.

➤ **To import MSDL:**

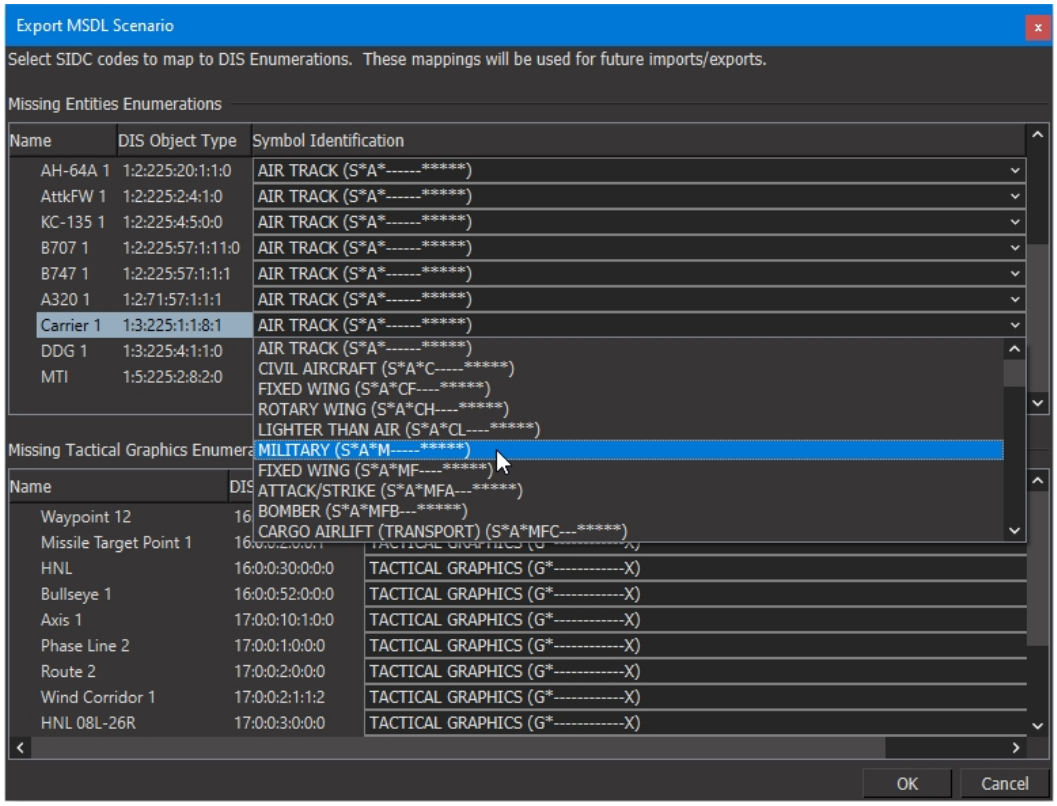
1. Create or load a scenario.
2. Choose **File > Import Scenario Objects**. The Import Scenario File dialog box opens.
3. In the Files of Type box, set the filter to MSDL Scenario File (*.xml).
4. Select the MSDL file that you want to import.
5. Click Open.

Exporting MSDL

➤ **To export a scenario to MSDL:**

1. Create or load a scenario.
2. Choose **File > Export Scenario Laydown**. The Export Scenario dialog box opens.
3. In the Save As Type box, set the filter to MSDL Scenario File (*.xml).
4. Select the directory in which you want to save the file.
5. Type a name for the file.
6. Click Save. If VR-Forces is not sure how to export an entity type to MSDL, the Export MSDL Scenario dialog box opens (Figure 63).

Figure 63. Export MSDL Scenario dialog box



7. For each object type listed in the Export MSDL Scenario dialog box:
    a. In the Symbol Identification column, click the value. A list of possible identification codes is displayed. (These codes are from the MSDL specification, MILSTD 2525B. You need to familiarize yourself with the specification to fully understand them.)
    b. Select the proper identification.
8. Click OK.

7.2.3 Importing Scenario Objects from CSV Files

Some VR-Forces users like to plot tactical graphics and simulation object locations offline and then import them into a scenario. VR-Forces supports importation of points, lines, and areas from comma separated values (CSV) files.

A CSV file can contain only one kind of object:

- One area.
- One route.
- One or more point objects. Points can represent tactical graphics, such as waypoints, or individual entities.

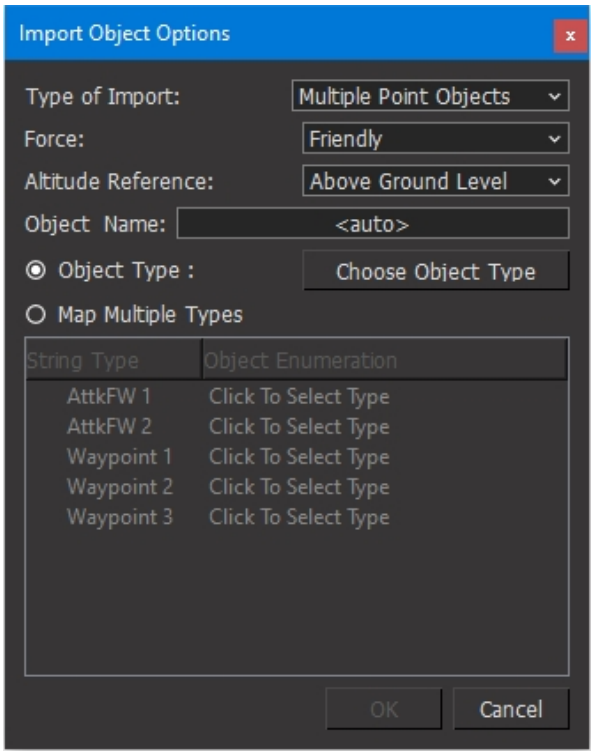
VR-Forces includes some example CSV files in *./userData/csvLaydown*.

> You cannot export simulation objects to a CSV file. To make the objects in a scenario available to other scenarios, try some of the strategies in "7.2. Importing Objects into Scenarios" on page 238.

➤ **To import scenario objects from a CSV file:**

1. Choose **File > Import Scenario Objects**. The Import Scenario File dialog box opens.
2. In the Files of Type box, set the filter to CSV Scenario File (*.csv).
3. Select the CSV file that you want to import.
4. Click Open. The Import Object Options dialog box opens. If you selected a CSV file that has point data, the points are listed in the window

Figure 64. Import Object Options dialog box

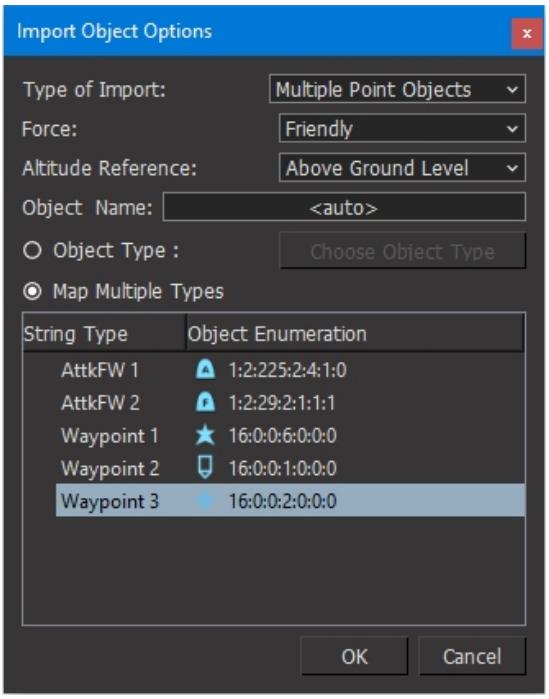


5. In the Type of Import, select the type of data you are importing.
6. Optionally, specify a force. All imported objects are imported with the same force.
7. Optionally, specify the altitude of objects as AGL (above ground level) or ASL (above sea level).
8. Optionally, in the Object Name box specify a name for the object. The default is to use VR-Forces naming conventions to automatically assign a name.

9. If you are importing an area or route:
    a. Select the Object Type option.
    b. Click Choose Object Type. The Entity Type dialog box opens. It lists objects of the type you selected in the Type of Import list. (The layout of the dialog box is similar to the object creation palettes.)
    c. Select the specific type of area or route that you want to use for the imported object. The Entity Type dialog box closes.

    If you are importing a list of waypoints:
    a. Select the Map Multiple Types option.
    b. Click a point in the list. The Object Type dialog box opens.
    c. Select the specific type of point or entity that you want to use for the imported object. The Object Type dialog box closes. The Import Object Options lists the object type enumeration for the object.

Figure 65. Import Object Options dialog box



    d. Repeat this process for each object in the list.
10. Click OK.

7.2.4 Importing Airspace Control Orders

An airspace control order (ACO) defines and establishes an airspace for military operations. It is can be part of an air tasking order (ATO) or a separate document. VR-Forces can import ACOs that are separate documents.

VR-Forces includes an example ACO file, *./userData/aco/exampleMontereyACO.aco*. The file's format is documented in *./userData/aco/acoFormattingReadme.txt*.

➤ **To import an ACO:**

1. Choose **File > Import Scenario Objects**. The Import Scenario File dialog box opens.
2. In the Files of Type box, set the filter to ACO Scenario File (*.aco).
3. Select the file that you want to import.
4. Click Open.

7.2.5 Importing Two-Line Element Sets

You can import two-line element set files (TLE) to display orbital objects, such as satellites.

> You cannot export simulation objects to a TLE file. To make the objects in a scenario available to other scenarios, try some of the strategies in "7.2. Importing Objects into Scenarios" on page 238.

➤ **To import a TLE:**

1. Choose **File > Import Scenario Objects**. The Import Scenario File dialog box opens.
2. In the Files of Type box, set the filter to TLE Scenario File (*.tle).
3. Select the file that you want to import.
4. Click Open. The Import Two-Line Elements dialog box opens.
5. Click Choose Object Type. The Object Type dialog box opens, showing available space simulation objects from the SMS you selected for the scenario.
6. Select the object you want to use for the imported objects. The Object Type dialog box closes.
7. Click OK.

7.2.6 Loading Object Tags

You can load a CSV file of entity types and tags. The tags are added to the currently loaded SMS of objects in the palettes but are not written back to the SMS. The next time you load the SMS, your custom tags are cleared out. This allows you to temporarily change the tags that are available in the palettes for filtering the list. There is an example file, *./SharedData/19/latest/ModelData/Configuration/modeltags/tags.csv*.

➤ **To load object tags into your scenario:**

1. Choose **File > Load Object Tags**. The Load Object Tags dialog box opens.
2. Navigate to the location of your CSV file and then select it.
3. Click Open.

   The updated tags are now available in your palettes.

7.3. Saving a Scenario

The first time you save a scenario, VR-Forces saves the path of the terrain database and other scenario configuration data to a scenario file (extension *.scn*). (Paths are saved relative to the *./bin64* directory.) This information usually does not change between runs of a scenario. Every time you save a scenario, VR-Forces saves the list of simulation objects and tactical graphics, the plans, and object mapping information to individual files that have the same name as the scenario, but with file extensions that identify the information in the file (*.oob*, *.pln*, *.xtr*, *.omp*, and so on). By default, all these files are saved to a compressed archive with the extension *.scnx*. However, you can save them as individual files, as explained in this procedure.



- If you want to keep the initial state of a scenario (simulation time = 0 and no simulation has taken place), so that you can repeatedly run a simulation from the beginning, save the scenario when you have finished creating it, but before you run it. Then, do not save the scenario again under the same name (unless you specifically want to change the original state of the scenario). (If you forget to save a new scenario, you can rewind to the initial snapshot and save it.)
- If you load an alternate terrain in a GUI when it joins a session, the alternate terrain is not saved as part of the scenario.

When you save a scenario, whether during the course of setting it up or during a simulation, all object state information is saved. Essentially, every save is a checkpoint. When you run a saved simulation, all simulation objects resume whatever they were doing when the scenario was saved.



- If simulation objects are executing scripted tasks, some information regarding the tasks is not saved. For details, see "37.2.3 Limitations for Checkpointing Scripts" on page 793.
- When you task an entity to move and then start playing the scenario, it generates a path. If you save the scenario with that path already planned, it is saved in the scenario. The next time you play the saved scenario, the entity does not need to calculate a path. This can improve the scenario performance.

Save behavior depends on when you save a scenario:

- If you save a previously saved scenario after it has started (whether it is paused or running), it is not saved to the original scenario file. It is saved to a checkpoint that is named using the convention *scenario_name*_at_time_*time*.scn. This ensures that you cannot inadvertently overwrite the original scenario file. To overwrite the original scenario, you must use Save Scenario As. For more information, see "7.8.1 Saving Checkpoints" on page 265 and "7.3.1 Saving a Previously Saved Scenario" on the next page.
- If you run a new scenario without saving it and save it at some simulation time greater than zero, it is saved using whatever name you provide. The time is not included in the scenario name.

➤ **To save a new scenario:**

1. Choose **File > Save Scenario** or click the Save button () on the File Toolbar. The Save Scenario dialog box opens.
2. Select the directory in which you want to save the scenario.

   On Windows machines, if you create a new folder using the Save Scenario dialog box, do not enter a space at the end of the name. Windows does not recognize the existence of folders with spaces at the end and it can cause problems. If you try to delete it, the system will not allow you to and it may blue screen. Running ScanDisk can solve this problem.

3. Give the scenario a name.
4. By default, scenarios are saved in a compressed archive (**.scnx*). If you want to save a scenario as individual files, in the File of Type list, select MAK Scenario Format (*.scn).
5. Click Save. All scenario files are saved with the new name.

   If you save a remapped scenario, you are prompted to save the scenario with the original mappings or with the remappings.

7.3.1 Saving a Previously Saved Scenario

In most computer applications, when you save a file that has been saved previously, the current version of the file simply overwrites the previously saved version. However, when VR-Forces saves a scenario, the save action depends on the status of the scenario. VR-Forces tracks the simulation time of the scenario. When you save a scenario:

- If a version of the scenario exists with the current simulation time, VR-Forces displays a message indicating that a version of the scenario at that simulation time exists and asks if you want to overwrite it.
- If there is no version with the current simulation time, VR-Forces creates a new checkpoint at the current simulation time, using the checkpoint naming convention.

This behavior ensures that if you change a scenario after you have started a simulation, you will never inadvertently overwrite the original version or another checkpoint.

➤ To save a previously saved scenario, choose **File > Save Scenario**.

Saving a Load Balanced Scenario

When you load a load-balanced scenario and do not have all of the required sim engines available, VR-Forces prompts you to allow it to remap objects to the available sim engines. Then it loads the scenario.

When you save a scenario that was originally load balanced, VR-Forces prompts you to either save the original mappings or to save the scenario with the sim engine mappings used to load it in the current session.

If you save the original mappings, all objects that were part of the scenario when it was loaded maintain their previous mappings. Any objects that you created during the current session are mapped to the sim engines that they were created on in the current session.

7.3.2 Saving a Scenario to a New Name or a Different Format

You can save a scenario to a different name. You can also save it to a different format. That is, you can save a scenario that was originally saved as a compressed archive to individual files or save individual files to a compressed archive.

If you save a scenario to a different format and you use the same name, VR-Forces does not replace the previously saved files with the new format. The two sets of files live side-by-side. Therefore we recommend that you give the newly saved scenario a different name or remove the old files to avoid confusion between different versions of the same scenario.

➤ **To save a scenario to a new name:**

1. Choose **File > Save Scenario As**. The Save Scenario dialog box opens.
2. Select the directory in which you want to save the scenario.
3. Give the scenario a name.
4. If you are changing the format, select an option in the Files of Type list.
5. Click Save. The scenario is saved to the new name.

7.4. Loading a Scenario

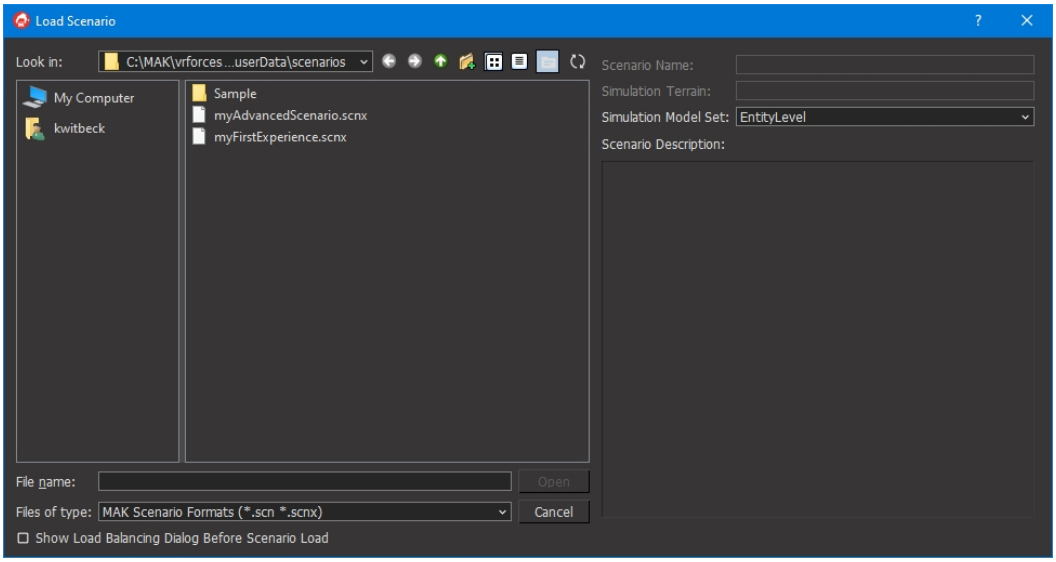
After you have created and saved a scenario, you can load it and run it on any platform supported by the version of VR-Forces in which you created it. Most scenarios are forward compatible. See the *VR-Forces Release Notes* for compatibility issues.

- If you are running multiple GUIs in the same session, when you load a scenario, any terrains that are open on other GUIs close and the terrain for the newly loaded scenario opens.
- When you load a scenario that uses multiple sim engines, if any of the required sim engines are not running, VR-Forces prompts you to remap the objects in the scenario. The Remap/Start Simulation Engines dialog box lists the unavailable sim engines and the available sim engines. You can specify which sim engines to map objects to or you can select "map all to" and map all objects to one sim engine. For more information about remapping sim engines, see "3.2.5 Working with Multiple Simulation Engines" on page 118.
- If the selected scenario uses a terrain that is not available, you are given the option to select another terrain. The message informs you that a different terrain may affect the simulation. It also recommends that the GUI Checks for Existence of Sim Terrain on Load check box is selected. That option is in the GUI Settings page of Application Settings.

➤ **To load a scenario:**

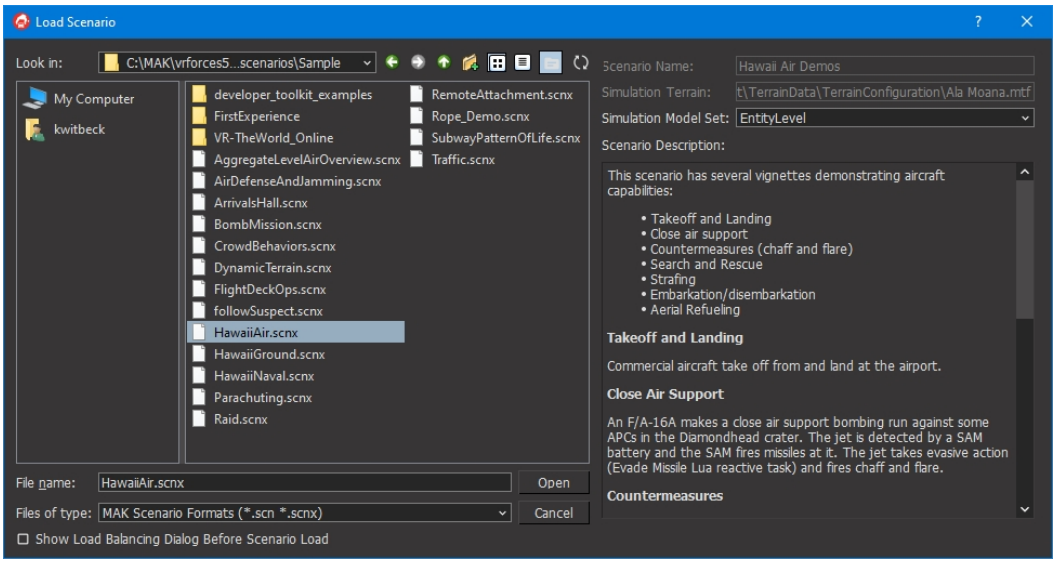
1. Choose **File > Load Scenario** or click the Open Scenario button () on the File Toolbar. The Load Scenario dialog box opens (Figure 66).

Figure 66. Load Scenario dialog box



2. Select a scenario to load. Basic information about the scenario displays (Figure 67). Scenarios are files with the suffix *.scn* or *.scnx*. Sample scenarios are located in the *./userData/scenarios/Sample* directory. (Batch scenarios, described in "7.10. VR-Forces and Batch Mode" on page 269, have the suffix *.bsn*.)

Figure 67. Load Scenario dialog box



3. Optionally, specify whether to allow management of components on remote entities by selecting an option in the Attach Components to Remote Entities On list. For details about configuring components on remote entities, see "82.1. Attaching

Components to Remote Entities" on page 1592.

4. Optionally, select the Show Load Balancing Dialog Before Scenario Load check box. See "7.5.1 Load Balancing an Existing Scenario on Load" on page 255 for details on the different options.
5. Click Open. If you elected to load balance the scenario, the Sim Engine Load Distribution dialog box opens. Otherwise, the scenario opens.

As the scenario loads, an indicator near the bottom right of the main window shows what is loading: terrain, insets, navigation, and so on.

7.4.1 Loading a Recently Loaded Scenario

VR-Forces keeps a list of the scenarios you open. You can quickly open a scenario from the list.

➤ To open a recently used scenario, choose **File > Recent Scenarios >** *scenario_name*.

7.4.2 Loading a Scenario from the Command Line

If you start VR-Forces from the command line, you can specify a scenario to load. You can specify the scenario in the sim engine or GUI. The other executable will load it as soon as it starts up.

To load a scenario when you start the sim engine, use the (`--scenarioFileName | -L`) command-line option, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --scenarioFileName
  "../userData/scenarios/myScenario.scnx"
```

To load a scenario when you start the GUI, use the `--scenarioFile` command-line option, for example:

```
vrfGui --scenarioFile "../userData/scenarios/myScenario.scnx"
  --dis --siteId 1 --appNumber 3000
```

7.4.3 Editing Scenario Properties

When you create a scenario, you specify a variety of parameters (or accept default values) and, optionally, specify a name and description for the scenario. If a scenario has a description, when you load the scenario, the Scenario Information dialog box opens automatically and displays the scenario properties. (You can disable this behavior.) You cannot edit properties in the Scenario Information dialog box.

You can display scenario properties at any time while a scenario is loaded. You can edit the scenario properties. These changes take place immediately:

- Scenario description.
- Scenario name.
- Time multiplier.
- Create global environment.
- Create global dynamic terrain processor.
- Scenario end time.

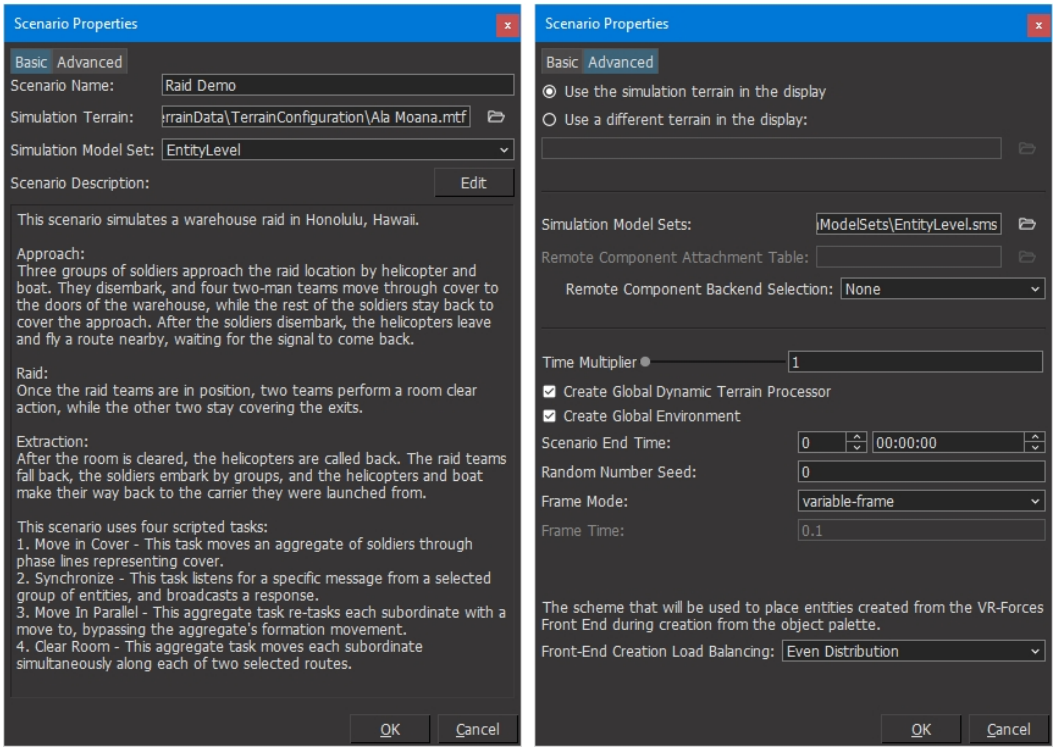
The following changes do not take effect until you save the scenario and reload it. If you edit one of these properties and have not yet saved the scenario, if you open the dialog box again, an icon displays to alert you to save the scenario.

- Simulation terrain.
- Use a different terrain in the display.
- Simulation model sets.
- Component attachment table.
- Random number seed.
- Frame mode.
- Frame time.

➤ **To edit scenario properties:**

1. Choose **File > Scenario Properties**. The Scenario Properties dialog box opens (Figure 68).

Figure 68. Scenario Properties dialog box



2. Edit properties as desired. (For descriptions of the properties, see Table 13 on page 230 and Table 14 on page 231.)

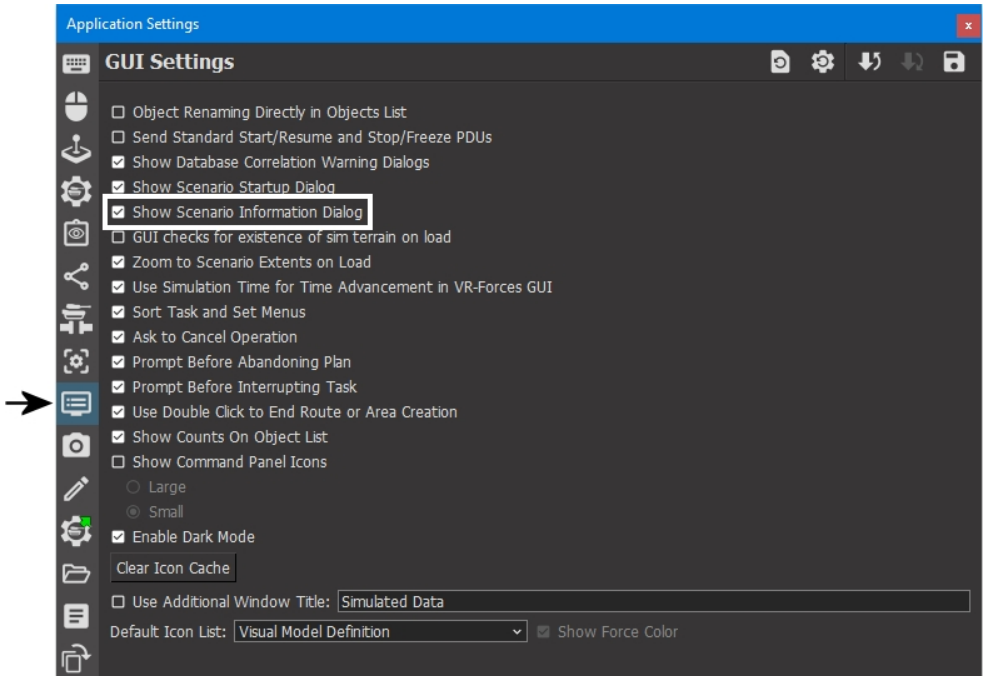
Enabling and Disabling Display of the Scenario Description

By default, the scenario description displays when you open a scenario. You can turn it off in the Scenario Information window and you can enable and disable its display in the Application Settings dialog box.

➤ **To disable display of scenario information when a scenario is loaded, use one of the following methods:**

- When the Scenario Information window opens, select the Never Show Again check box.
- On the Application Settings dialog box, GUI Settings page, clear the Show Scenario Information Dialog check box (Figure 69).

Figure 69. GUI Settings page



➤ To enable the Scenario Information dialog box, on the Application Settings dialog box, GUI Settings page, select the Show Scenario Information Dialog check box.

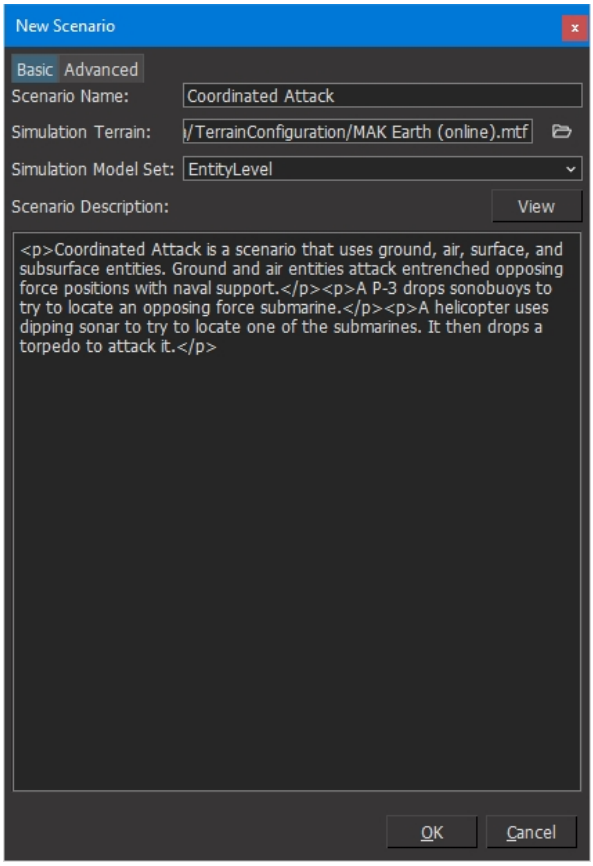
7.4.4 Editing the Scenario Description

You can edit a scenario's description in the Scenario Properties dialog box. If you want to format it, you can use HTML tags.

➤ **To edit a scenario's description:**

1. Choose **File > Scenario Properties**. The Scenario Properties dialog box opens (Figure 70).

Figure 70. Scenario Properties



2. Edit the text.
3. To see how the text will be displayed, click View.
4. Click OK.

7.4.5 Navigating Scenario Directories

When you use the Load Scenario dialog box, the default ("home") directory is *./userData/scenarios*. You can change the default directory using the --userScenarioDirectory *directory* command-line option.

If you open a scenario from another directory, the next time in that session that you open the Load Scenario dialog box, it defaults to that directory. If you want to jump to the home directory, click the Return to Default Scenario Directory button (🔄).

7.4.6 Sample Scenarios

VR-Forces includes a number of sample scenarios. The *VR-Forces First Experience Guide* has several tutorials for scenario creation.

7.5. Load Balancing Scenarios

The preferred method for optimizing the performance of scenarios is to manage multiprocessors usage as described in "Optimizing Multiprocessor Usage for Simulation Engine" on page 197.

If configuring VR-Forces to take advantage of your specific hardware setup does not improve the performance of a scenario, you can try spreading simulation of objects across multiple sim engines and machines. This is called load balancing. VR-Forces supports even distribution (done automatically by VR-Forces) and manual load balancing.

You can load balance a scenario when you create it. You can also load balance a scenario after you create it.

➤ **To load-balance a new scenario:**

1. Start multiple sim engines on separate machines as described in "4.1.1 Starting VR-Forces from the VR-Forces Launcher" on page 129 and "4.1.2 Starting Independent VR-Forces Executables" on page 132 .
2. Create the new scenario, as described in "7.1. Creating a Scenario" on page 228.
3. Before you create an object, select the sim engine on which you want it to be simulated, as described in "14.2.1 Selecting the Object from the Create Object Palette" on page 374 and "14.2.2 Selecting the Object to Create on the Create Menu" on page 377.

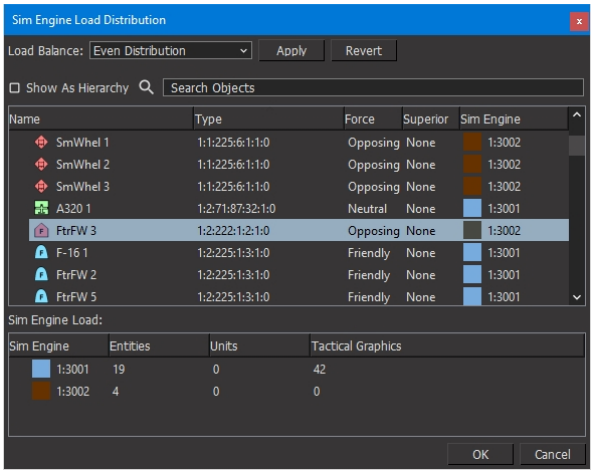
7.5.1 Load Balancing an Existing Scenario on Load

You can change the object mappings for a scenario (load balancing) when you load it. You cannot change mapping after the scenario loads.

➤ **To load-balance a existing scenario when you open it:**

1. Start as many sim engines as you want to use in the scenario.
2. Choose **File > Load Scenario**. The Load Scenario dialog box opens.
3. Select the Show Load Balancing Dialog Before Scenario Load check box.
4. Click Open. The Sim Engine Load Distribution dialog box opens (Figure 71).

Figure 71. Sim Engine Load Distribution when loading a scenario



5. Choose an automatic load balance method:
   - Even Distribution. VR-Forces automatically distributes objects across the available sim engines.
   - Unit Based Distribution. VR-Forces ensures that aggregates and subordinates are kept together on the same sim engine.
   - Rule Based Distribution. Choose an existing rule set or create a new one. See "Rule-Based Load Balancing" below to learn about creating a rule set.

   You can also load balance the scenario manually. See "Manually Load Balancing a Scenario" on the facing page.
6. After selecting the load balance distribution method, click Apply. The Sim Engine column of the list shows which sim engine each simulation object will load on, and the Sim Engine Load shows the totals.
7. Click OK. The scenario loads. If you open the Information dialog box for an object, the State Data page shows the Simulating Sim Engine.
8. To preserve the new object mappings, save the scenario. When you save the scenario, you are prompted to save the original mappings. Click No to save the new load balanced mappings.

Rule-Based Load Balancing

You can create a new rule set for load balancing. This allows you to specify where simulation objects should load based on object type or name.

➤ **To create a rule set:**

1. Enter a Title for the rule set.
2. Click the down arrow next to the plus sign and then choose either Entity Type or Object Name.

- When you choose Entity Type, it adds a row. The default is all entities. Click Edit ( ) and then use the Edit Entity Type dialog box to specify particular simulation objects.
- When you choose Object Name, the Object Name dialog box opens. Enter a specific simulation object's name or create a regular expression.

3. Add any other rules that you want.
4. When you are finished, click OK.

Manually Load Balancing a Scenario

When loading a scenario, you can manually move simulation objects from one sim engine to another.

➤ **To manually transfer ownership of a simulation object:**

1. In the Sim Engine Load Distribution dialog box (Figure 71), right-click on the simulation object.
2. If it is a individual simulation object, select the sim engine where you want it to be simulated.

   If it is a unit, choose whether to move the unit alone (Self) or the unit and the members that comprise it (Self and Subordinates) and then select the sim engine.

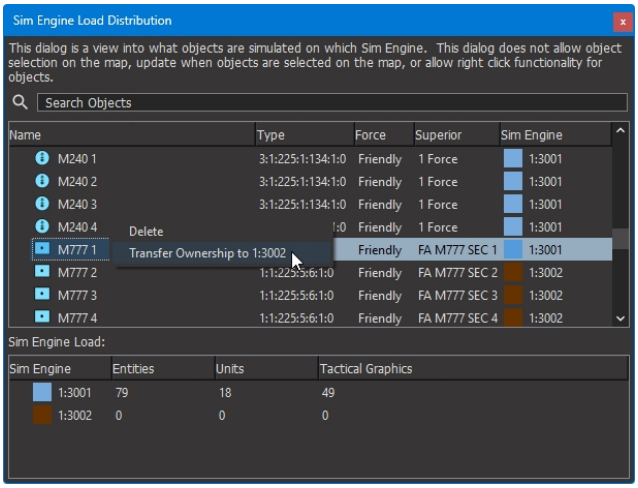
To learn about moving simulation objects to another sim engine after loading a scenario, see "7.5.2 Load Balancing After Loading the Scenario" below.

7.5.2 Load Balancing After Loading the Scenario

The procedure for moving entities from one sim engine to another after you load a scenario is similar to doing it beforehand.

At this time, ownership transfer is supported only in HLA Evolved and HLA 4 and you can only move entities. You cannot move units or aggregates.

Figure 72. Sim Engine Load Distribution for loaded scenarios



➤ **To transfer ownership of entities after loading a scenario:**

1. Choose **Simulation > Load Distribution**. The Sim Engine Load Distribution dialog box opens.
2. In the list of simulation objects, right-click on the entity you want to move and then select the desired sim engine. The entity moves and the list shows the sim engine that now has ownership.

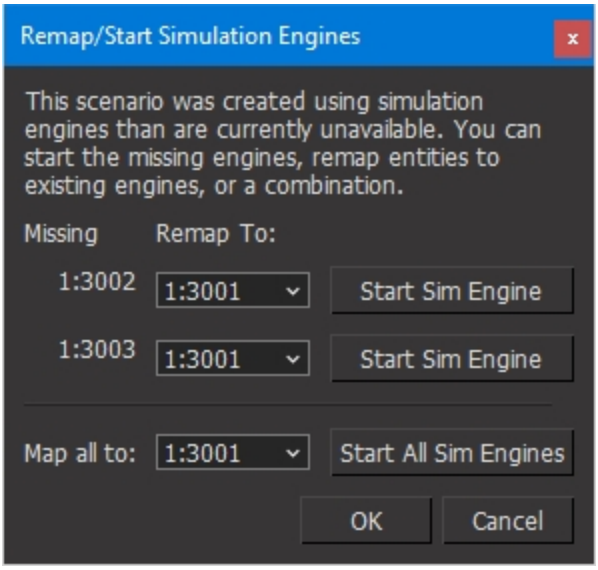
   You can use **Shift**+click or **Ctrl**+click to select multiple entities. After selecting them, right-click on one of the selected ones and then select the sim engine.

3. When you are finished, close the dialog box.

7.5.3 Loading a Load-Balanced Scenario

When you load-balance a scenario, it saves the mapping between objects and simulation engines in the scenario's object mapping file. When you load the scenario, VR-Forces looks for the listed simulation engines based on their application numbers. If it does not find the required simulation engines, it prompts you for instructions on how to proceed (Figure 73). You can start the missing simulation engines, remap objects to available simulation engines, or if you are using more than two simulation engines, do a combination of both.

Figure 73. Remap/Start Simulation Engines dialog box



If you start a simulation engine as part of the process of loading a scenario, it is started in combined mode and will automatically be closed when you shut down the GUI.

➤ **To load a scenario that uses multiple simulation engines:**

1. Optionally, start all the required simulation engines and at least one GUI. One simulation engine can be started in combined mode with the GUI.
2. Load the scenario. If the scenario cannot find all of the required simulation engines, it prompts you to start them or remap objects to the available sim engines (Figure 73). For each unavailable simulation engine, you can remap objects to an available simulation engine or start the simulation engine:
   - To start a simulation engine, click the Start Sim Engine button for the Missing simulation engine.

     If there is more than one unavailable simulation engine, as you start them individually, the dialog box redisplays to show only those that are still unavailable. If you start all of the required simulation engines, the dialog box closes and the scenario loads.
   - To start all missing simulation engines, click the Start All Sim Engines button.
   - To remap objects for a missing simulation engine, select an available simulation engine from the Remap To list for the Missing simulation engine.
   - To remap all objects to an available simulation engine, select an available simulation engine from the Map All To list.

     When you select a simulation engine in the Map All To list, it sets the same simulation engine in the lists for each specific missing simulation engine. Put another way, when you click OK to close the dialog box, VR-Forces uses the

remapping specified for each missing simulation engine. It ignores the setting for Map All To. Map All To is just a quick way to set all missing simulation engines to the same available one.

3. If you decided to remap objects to an available simulation engine, click OK. VR-Forcess remaps the objects and the scenario loads.

7.6. Running a Scenario

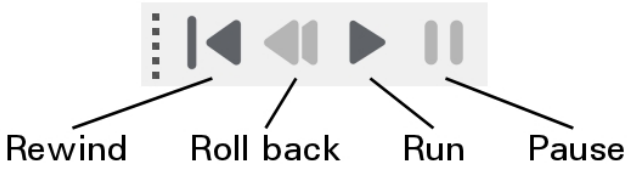
If you want to use VR-Forces to simply observe a remote simulation, you can start it, open a terrain database, and wait for it to pick up messages from the network. However, you will probably use VR-Forces to run scenarios that have locally simulated simulation objects.

When you load a scenario, it opens in the paused state unless you start with the `-L` and `-r` sim engine command line options.

> When you run a scenario, VR-Forces saves a snapshot of the state of the scenario in memory. However, it does not save the scenario to disk. We recommend that you save new or edited scenarios before you run them. If you choose not to, it is your responsibility to remember to rewind to the initial scenario snapshot if you want to save the starting point of the scenario.

➤ To run a scenario, choose **Simulation > Run Scenario**, or click the Run button (▶) on the Simulation Control Toolbar (Figure 74).

Figure 74. Simulation Control Toolbar



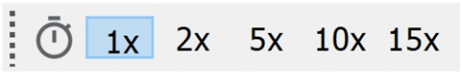
7.6.1 Changing the Simulation Speed

You can set the speed of a simulation to run faster or slower than real-time. If you are running VR-Forces with time management, the Simulation Time Scale Toolbar has no effect on simulation speed.

> ⚠ If you increase the speed at which a scenario runs, the frame rate is reduced and performance of models may degrade.

➤ To change the speed of a simulation, click an option on the Simulation Time Scale Toolbar (Figure 75).

Figure 75. Simulation Time Scale Toolbar



Configuring the Simulation Time Scale Toolbar

By default, the range for the Simulation Time Scale Toolbar is 1 to 15. You can change the range. To change the range, edit these lines in *./appData/settings/vrfGui/default_GuiSettings.grsx*:

```
<myTimescaleLow>1</myTimescaleLow>
<myTimescaleHigh>15</myTimescaleHigh>
```

To change the options on the time scale buttons, edit the myTimescaleButtons elements:

```
<myTimescaleButtons>1;2;5;10;15</myTimescaleButtons>
```

If you want the Simulation Time Scale Toolbar to use a slider as in VR-Forces 4.6.1 and earlier, instead of discrete buttons, set the myTimescaleButtonMode to 0:

```
<myTimescaleButtonMode>0</myTimescaleButtonMode>
```

Changing Simulation Parameters when Increasing the Time Scale

When you increase the time scale in scenarios that have many entities, particularly if they are fast moving entities such as aircraft, simulation quality may degrade. VR-Forces lets you change the dead-reckoning algorithm, frame mode, and frame time automatically to settings that may improve simulation quality.

These settings are configured in *./appData/settings/vrfSim/fastForwardSettings.mtl*. The comments in the file explain the acceptable values for each parameter. This is an example configuration:

```
(fast-forward-settings
  (fast-forward-entry
    (play-speed 3)
    (settings
      (dead-reckon-algorithm Static)
      (frame-mode 2)            ;; fixed frame
      (frame-time 1.0)
    )
  )
  (fast-forward-entry
    (play-speed 6)
    (settings
      (dead-reckon-algorithm Static)
      (frame-mode 2)            ;; fixed frame
      (frame-time 2.0)
    )
  )
)
```

Each fast-forward entry specifies a time scale speed at which it takes effect. When you change the time scale, VR-Forces finds the highest play-speed specified in this file that is less than the current speed and updates the settings accordingly. If the current speed is less than the lowest value, default settings are used.

7.6.2 Pausing a Scenario

When you pause a simulation, local simulation objects stop executing plans and tasks. Display of remote entity activity continues as it is received over the network.

➤ To pause a simulation, choose **Simulation > Pause Scenario**, or click the Pause button ( ) on the Simulation Control Toolbar (Figure 75 on page 260).

7.6.3 Rewinding a Scenario

If you are running a scenario and want to start it over again, you can rewind it. You can rewind a scenario to its initial state, which is captured as a snapshot, and you can roll back to any snapshot or checkpoint (if snapshots and checkpoints are enabled).

When you rewind a scenario, all sim engines are set to the paused state.

> An initial snapshot is taken of a scenario when you play it even if periodic snapshots is not enabled. The initial snapshot is used for rewinding.

➤ To rewind a scenario, choose **Simulation > Rewind to Scenario Start**, or click the Rewind Scenario to the Beginning button ( ) on the Simulation Control Toolbar (Figure 75 on page 260).

If you do not pause a scenario before you rewind it, it starts playing again immediately.

Rolling Back to a Snapshot or Checkpoint

You can roll back to any of the available snapshots or checkpoints. When you roll back to a snapshot, VR-Forces loads the scenario as it existed at the point of that snapshot. When you run the snapshot, all snapshots from a later time than the one you rolled back to are deleted. Older snapshots are kept in memory. New snapshots are taken starting from the loaded snapshot.

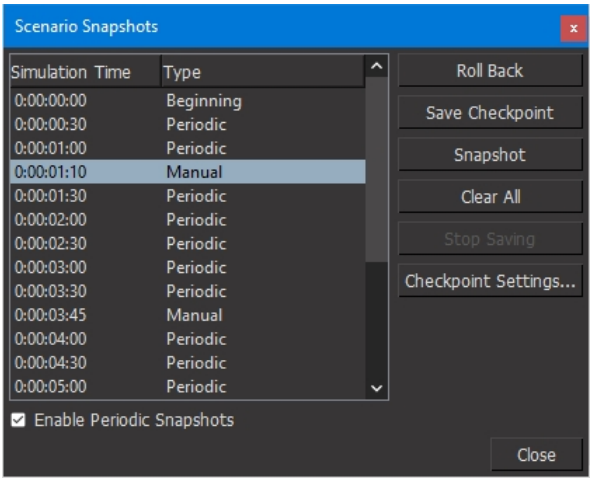
Once snapshots are deleted based on the maximum number of snapshots being reached, you cannot roll back to them. If you have saved periodic checkpoints, you can load them using the Load Scenario command on the File menu.

For details about checkpoints and snapshots, see "7.8. Checkpoints and Snapshots" on page 264.

➤ **To roll back to a snapshot or checkpoint:**

1. Choose **File > Scenario Snapshots**, or on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 76).
2. Select the snapshot or checkpoint you want to roll back to.

Figure 76. Scenario Snapshots dialog box

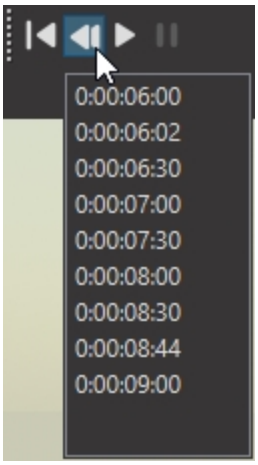


3. Click Roll Back.

Using the Snapshot and Checkpoint List

You can also display a list of snapshots and checkpoints by clicking the Snapshots button () on the Simulation Control Toolbar (Figure 77).

Figure 77. Snapshot list



7.6.4 Closing a Scenario

When you close a scenario, the terrain stays loaded. Checkpoints and snapshots remain in memory until a new scenario is loaded. Therefore, if you click Rewind after closing a scenario, the scenario opens again.

➤ To close a scenario, choose **File > Close Scenario**, or click the Close Scenario button () on the File Toolbar.

7.7. Reinitializing a Scenario

Sometimes when you load a scenario that was created in a previous version of VR-Forces, entities do not behave as they did previously due to changes to systems. Usually, the way to fix this problem is to delete the objects, create new instances of the object type, and reconstruct their plans or tasks. Unfortunately, when you create a new object, it is assigned a unique ID. Since the unique ID for the newly created objects will be different from those of the deleted objects, any plans or tasks that referenced the deleted objects will not work, because they depend on the unique ID to identify the object that they are referencing. If you need to recreate an object and maintain its unique ID, you can reinitialize a scenario.

When you reinitialize a scenario, VR-Forces deletes all simulation objects and creates them again with the same unique IDs they originally had. This preserves task and plan relationships. However, this process does not preserve the entire state of entities. If you reinitialize a checkpointed scenario, it may lose information that is saved as part of a system. Plans are restarted, so the current task might be cleared. So there may be cases where reinitializing causes more problems than it solves.

Given this consideration, you should only use this option for scenarios that are at zero simulation time and only if you are sure that it is the only option. You may want to save a backup of the scenario in case you want to revert to it and try fixing it by manually deleting the offending object and adjusting other aspects of the scenario as needed.

➤ **To reinitialize a scenario:**

1. Optionally, save a backup of the scenario in case you want to revert to the original state.
2. Choose **File > Reinitialize Scenario**.

7.8. Checkpoints and Snapshots

As noted in "7.3. Saving a Scenario" on page 247, each time you save a scenario, you are essentially creating a checkpoint. A checkpoint saves the state of a scenario to disk and allows you to run the scenario starting from the point at which it was saved. Since a checkpoint is saved to disk, you can return to it at any time. However, the time spent saving to disk may briefly interrupt scenario progress.

Sometimes VR-Forces users want to be able to quickly return to an earlier point in a simulation without the processing overhead required to save checkpoints. As an alternative or complement to checkpoints, you can save snapshots.

Snapshots save the state of a scenario to memory. You can quickly roll back to any saved snapshot and start running the scenario from that point. When you roll back to a snapshot, all snapshots that were taken at a later time than the rollback point are discarded and new snapshots are saved. When you load a scenario, any snapshots in memory are discarded.

You can configure the frequency with which snapshots are saved and how many to keep in memory at a time. You can save snapshots as checkpoints.

7.8.1 Saving Checkpoints

To save checkpoints, you can :

- Save the scenario.
- Save a snapshot as a checkpoint.
- Enable periodic checkpointing.

Checkpoints are saved to the same directory as the base scenario unless you specify otherwise. If periodic checkpoints are enabled, the Status bar displays the time until the next checkpoint.

> If simulation objects are executing scripted tasks, some information regarding the tasks is not saved. For details, see "37.2.3 Limitations for Checkpointing Scripts" on page 793.

When you run a checkpoint, simulation objects resume the tasks that they were executing at the time the checkpoint was saved. If detonations were scheduled or if missiles were in flight, they resume their former state.

Saving an Individual Checkpoint

Whenever you save a scenario, it creates a checkpoint.

➤ To save an individual checkpoint, choose **File > Save Scenario**.

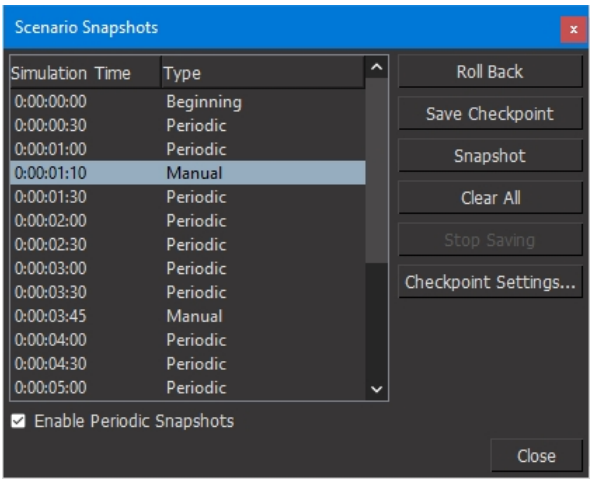
Saving Snapshots as Checkpoints

If you enable periodic snapshots, you can save the snapshots as checkpoints.

➤ **To save a snapshot as a checkpoint:**

1. Choose **File > Scenario Snapshots**. The Scenario Snapshots dialog box opens (Figure 78). It lists all of the snapshots available.

Figure 78. Scenario Snapshots dialog box



2. Select the snapshots that you want to save as checkpoints.
3. Click Save Checkpoint. The snapshots are saved to checkpoints.

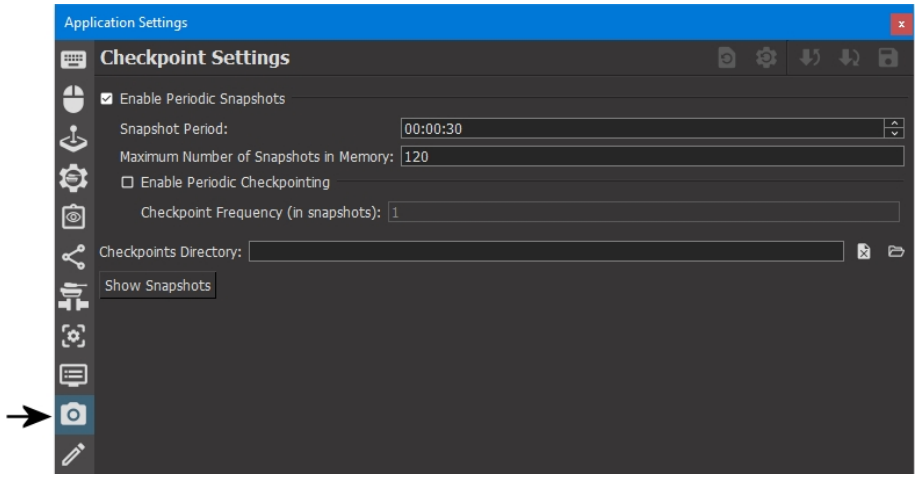
Scheduling Periodic Checkpoints

You can configure VR-Forces to save checkpoints at regular intervals. When VR-Forces saves a checkpoint, it is actually saving the snapshot taken at the checkpointing interval. Therefore, to schedule periodic checkpoints, you must enable periodic snapshots.

➤ **To schedule checkpoints:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 79).

Figure 79. Checkpoint Settings page



3. Select the Enable Periodic Snapshots check box.
4. In the Snapshot Period box, specify how frequently VR-Forces should take snapshots.
5. Select the Enable Periodic Checkpointing check box.
6. In the Checkpoint Frequency box, specify how frequently to save checkpoints as a function of the number of snapshots taken. For example, 4 means save every fourth snapshot as a checkpoint. If you save snapshots every 30 seconds, then you would save a checkpoint every two minutes.
7. Optionally, specify the directory in which to save the checkpoints. If you leave this box blank, they are saved to the directory where the base scenario is located.

Disabling Periodic Checkpointing

➤ **To disable periodic checkpointing:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 79).
3. Clear the Enable Periodic Checkpointing check box.

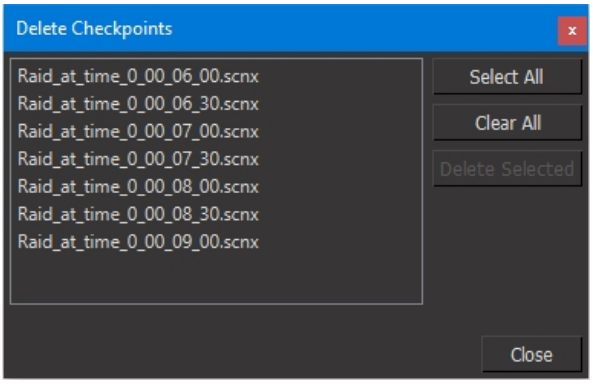
7.8.2 Deleting Checkpoints

You can delete checkpoints manually in a file browser, like you can delete any other files on your computer. You can also easily delete them from the GUI.

➤ **To delete checkpoints:**

1. Choose **File > Delete Scenario Checkpoints**. The Delete Checkpoints dialog box opens (Figure 80).

Figure 80. Delete Checkpoints dialog box



2. Select the checkpoints that you want to delete.
3. Click Delete Selected.

7.8.3 Creating Snapshots

You can create snapshots manually and you can specify that they be created automatically at a specified interval.

Creating Snapshots Manually

➤ **To create a snapshot:**

1. Choose **File > Scenario Snapshots**, or on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 78 on page 265).
2. Click Snapshot.

> You can also create a snapshot by clicking the Snapshot button ( ) on the Scenario Toolbar.

Enabling Periodic Snapshots

When you enable periodic snapshots, you specify how frequently to take them, in simulation time, and the maximum number to store in memory at a time. If the number of snapshots reaches the maximum, VR-Forces drops the oldest snapshot and adds the newest.

➤ **To enable periodic snapshots:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Checkpoint Settings page (Figure 79 on page 266).
3. Select the Enable Periodic Snapshots check box.
4. In the Snapshot Period box, specify how frequently VR-Forces should take snapshots.
5. In the Maximum Number of Snapshots in Memory box, specify how many snapshots to keep in memory.

7.8.4 Clearing the Snapshot List

Snapshots are stored in memory and listed in the Scenario Snapshots dialog box. They are cleared automatically when you load a scenario or rewind a scenario and run it again. If you roll back to a snapshot and run the scenario, all snapshots later than the rollback point are deleted. You can also clear snapshots manually.

When you clear the snapshot list, VR-Forces keeps the initial snapshot so that you can still rewind the scenario.

➤ **To clear the snapshot list:**

1. Choose **File > Scenario Snapshots** or, on the Checkpoint Settings page, click Show Snapshots. The Scenario Snapshots dialog box opens (Figure 78 on page 265).
2. Click Clear All.

7.9. Pausing a Scenario Automatically

You can specify a scenario end time (in simulation time). As the scenario progresses, the amount of time remaining is displayed in the Simulation Time Remaining Toolbar (Figure 81). When the scenario reaches this time, it stops. You can resume the scenario by clicking the Play button.

Figure 81. Simulation Time Remaining Toolbar



You can set the scenario end time when you create a scenario or after you have loaded it.

7.9.1 Setting the Scenario End Time for a New Scenario

The default scenario end time for new scenarios is 0 – do not pause the scenario.

➤ **To set the scenario end time for a new scenario:**

1. Create a new scenario as described in "7.1. Creating a Scenario" on page 228.
2. In the New Scenario dialog box, select the Advanced tab.
3. In the Scenario End Time boxes, specify an end time in days, hours, minutes, and

seconds.

4. Continue the scenario creation procedure.

7.9.2 Setting the Scenario End Time for an Open Scenario

You can set the scenario end time for a scenario after you open it. If you set a time that is sooner than the current elapsed time, the scenario stops immediately.

➤ **To set the scenario end time for an open scenario:**

1. Choose **File > Scenario Properties**. The Scenario Properties dialog box opens.
2. Select the Advanced tab
3. In the Scenario End Time boxes, specify an end time in days, hours, minutes, and seconds.
4. Click OK.
5. Optionally, save the scenario. Remember that if the scenario is already running or has advanced and then paused, the end time is saved to a checkpoint, not the original scenario.

7.10. VR-Forces and Batch Mode

Batch mode allows you to run one or more scenarios multiple times, without direct action through the graphical user interface. A scenario run in batch mode runs with the parameters in the scenario file, except that you can override the random number seed.

Batch mode is read-only. You cannot save a scenario, create simulation objects or objects, pause the scenario, or otherwise change it, while a batch is running. However, you can view the Objects List Panel, plan windows, and open informational dialog boxes. You can close a scenario that is running in batch mode and resume control of VR-Forces.

7.10.1 The Batch File

A batch mode file contains the information that VR-Forces needs to run scenarios in batch mode. Batch mode offers these options:

- Run a particular scenario multiple times.
- Run a series of different scenarios (batches), each of which can be run multiple times.
- Run from the command line using one sim engine. If a scenario was created using multiple sim engines, you cannot run it in batch mode using this method.
- Run from the VR-Forces GUI using one or more sim engines.
- Generate Logger Control messages to have the MAK Logger record each simulation.

For each batch (one scenario run one or more times) within the batch file, you can:

- Specify the length of time the simulation is to run (in real time or simulation time).
- Specify the object parameter database to use.

- Specify that the random number seed used be the value specified in the scenario file, a value specified in the batch file, or be computer generated.

A batch file has the file extension *.bsn*.

7.10.2 Creating a Batch File

To run VR-Forces in batch mode, you create a batch file (*.bsn*). VR-Forces supplies a sample batch file. Use it as a model for creating your own batch files.

➤ **To create a batch file:**

1. Create scenarios as you would any scenario. (For details, see "7.1. Creating a Scenario" on page 228.)
2. In a text editor, open the sample batch file provided with VR-Forces.
3. Save the batch file under a new name, with the file extension *.bsn*.
4. Replace the sample values with your own, as described in "7.10.3 Editing a Batch File" below.
5. Save the new batch file.

7.10.3 Editing a Batch File

A batch file is written in MAK Technologies Lisp format (MTL). (For information about the MTL format, see "B.2. MAK Technologies Lisp (MTL) Files" on page 1664.) A batch file starts with parameters that affect the entire batch. Then it has a batch list, which contains one or more batches. Each batch specifies a scenario, the number of times it is to be run, and other parameters. Table 15 describes the parameters.

This is an example of a batch file:

```
(scenario-batch
  (use-logger-control True)
  (logger-files-path "..\logger_tapes") ;; path is relative to
       Logger executable.
  (batch-list
    (batch
      (number-of-runs  2)
      (random-number-seed 4)
      (scenario-filename  "..\data\scenarios\Raid.scnx")
      (sms-filename
        "..\data\simulationModelSets\EntityLevel.sms")
      (simulation-run-duration 0 : 0 : 0 : 30.000000)
       ;; simulation time
    )
    (batch
      (number-of-runs  3)
      (scenario-filename
        "..\data\scenarios\Apache.scnx")
        (sms-filename
        "..\data\simulationModelSets\EntityLevel.sms")
```

```
      (run-duration 0 : 0 : 0 : 20.000000) ;; real time
    )
  )
)
```

Table 15. Batch file parameters

Parameter	Description
Scenario-batch parameters	
use-logger-control	Specifies whether to generate Logger Control messages. Options: • False – Do not generate messages. • True – Generate messages to create Logger files for each run of each batch.
logger-files-path	If **use-logger-control** is True, specifies the directory in which to store Logger files. The path can be absolute or relative to the Logger executable.
Batch-list parameters	
number-of-runs	Specifies how many times to run the scenario in this batch run entry.
random-number-seed	Optional. Overrides the random number seed in the scenario file. It specifies a random number seed: • -1 – use a value generated by the computer. Values are different for each run of this batch. • Any non-negative integer – use the number specified for each run of the batch.
scenario-filename	Specifies the scenario to be run in this batch run entry. The scenario can be specified as an absolute path or a path relative to the directory in which the executable is located.
sms-filename	Optional. Overrides the simulation model set specified by the scenario and specifies the simulation model set file to use.

Table 15. Batch file parameters (continued)

Parameter	Description
simulation-run-duration	Specifies the duration in simulation time that the simulation will run. It is mutually exclusive with **run-duration**. If both are specified, VR-Forces uses **simulation-run-duration**. ⚠ Specify hours, minutes, and days as integers, for example 15. Specify seconds as real numbers, for example, 15.0.
run-duration	Specifies the duration in real time that the simulation will run. For example, if a scenario has a time scale set to 2 (twice real time), and a run-duration of 10 minutes, 20 minutes of simulation time will elapse before the batch stops running this scenario. It is mutually exclusive with **simulation-run-duration**. ⚠ Specify hours, minutes, and days as integers, for example 15. Specify seconds as real numbers, for example, 15.0.

7.10.4 Running VR-Forces in Batch Mode

Running in Batch Mode in the GUI

➤ **To run VR-Forces in batch mode in the GUI:**

1. Start VR-Forces. If any of the scenarios you plan to run in batch mode require multiple sim engines, start the sim engines with the required site and application IDs.
2. If you are going to generate Logger control messages, start the MAK Data Logger. (Make sure to use the same protocol as you are using for VR-Forces.)
3. Choose **File > Load Batch File**. The Load Batch File dialog box opens.
4. Select the batch file you want to run.
5. Click Open.

VR-Forces begins running the first scenario in the batch file. When VR-Forces finishes running the batch file, it closes the final scenario.

ⓘ
- When VR-Forces runs a batch file, if it cannot load a scenario specified in the file, it skips the scenario and goes to the next batch entry in the file.
- Automatic remapping is always enabled for batch mode.

Running in Batch Mode from the Command Line

If your scenarios only need one sim engine, you can run a batch file from the command line.

➤ To run a batch from the command line, use the (--batchScenarioFileName | -B) parameter, with the sim engine in separate mode, for example:

```
vrfSimDIS --siteId 1 --appNumber 3001 --batchScenarioFileName
    "../userData/scenarios/Sample/sampleScenarioBatch.bsn"
```

The scenario begins to run automatically.

7.10.5 Recording Batch Scenarios

You can generate Logger Control messages and save scenario output to a Logger file. You must start the Logger before you run the batch scenario.

If you generate Logger Control messages, each simulation is saved individually by the Logger, with a filename in the format:

```
<batch_scenario_name>_<scenario_run_number>_<random_number_seed>.lgr
```

You can specify the directory in which you want the Logger file to be saved.

The MAK Data Logger does not record batch file runs if a previous recording exists. By default, the Logger does not allow you to overwrite existing files using the remote control API. Therefore, once you record a batch run, you cannot record a subsequent run of the same batch. You can work around this problem by editing the *lgrConfig.xml* file to permit overwriting of Logger files. Make the following edits in the configuration file for each protocol for which you want to record batch runs:

Add these lines to the <defaults> section:

```
<var name="processRemoteControl" type="bool" value="true"/>
<var name="recordRemoteControl" type="bool" value="true"/>
<var name="allowDeleteRemoteControl" type="bool" value="true"/>
```

Add these lines to the <commands name="startup"> section:

```
<command domain="Simulation" protocol="DIS"
command="SetRemoteControlSettings">
    <param name="process" var="processRemoteControl"></param>
    <param name="record" var="recordRemoteControl"></param>
    <param name="allowDelete" var="allowDeleteRemoteControl"></param>
</command>
```

Use the appropriate value for the protocol attribute. For DIS, set listenWhenIdle to false:

```
<var name="listenWhenIdle" type="bool" value="false"/>
```

7.11. Sending Run and Pause Messages to Simulation Participants

By default, when you run or pause a scenario, VR-Forces sends scenario control messages that are meaningful only to VR-Forces applications that are part of the current session. Sometimes VR-Forces users want all of the participants in a simulation, which may include non-VR-Forces applications, to know when VR-Forces runs or pauses a scenario. You can

achieve this by telling VR-Forces to send standard Start/Resume and Stop/Freeze messages instead of the VR-Forces-specific messages. All of the applications that are listening for them will receive them and can respond as needed.

> If you are running multiple VR-Forces sessions and you use Start/Freeze messages, all of the sessions receive them, which may defeat the purpose of having separate sessions.

➤ **To enable use of Start/Freeze messages:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 69 on page 253).
3. Select the Send Standard Start/Resume and Stop/Freeze PDUs check box.

7.12. Recording VR-Forces Simulations with the MAK Data Logger

You can use the MAK Logger to record VR-Forces simulations and then play them back for review. If you want to view a Logger file in VR-Forces (as opposed to some other MAK viewer, such as VR-Vantage), note the following considerations:

- Do not load the original scenario in VR-Forces while you view the recording. Just load the terrain database that the simulation requires.
- If you are using HLA 1516, do not record a simulation and then immediately play it back. Exit the applications and start a new federation. Trying to play back the file in the federation from which it was recorded can result in name reservation conflicts and applications may crash.

If a scenario includes plans that have view control messages, the Logger records them. When you play the recording, VR-Forces and other VR-Vantage applications will change view modes in response to them. This feature lets you easily create demonstrations that change views.



8. Scenario Events

Scenario events let you provide non-simulated input to scenario participants.

8.1. Introduction to Scenario Events 276
8.2. Creating a Scenario Event 276
8.2.1 Adding Content to a Scenario Event 279
8.2.2 Adding Linked Events 280
8.3. Adding Intelligence Objects 281
8.4. Starting a Scenario Event 283
8.4.1 Controlling Scenario Events from a Plan 284
8.5. Ending a Scenario Event 285
8.6. Enabling and Disabling Scenario Events 286
8.7. Sorting the Event List 287
8.8. Deleting a Scenario Event 287
8.9. Showing and Hiding Scenario Events 287

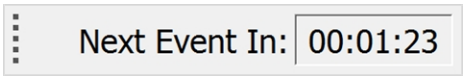
8.1. Introduction to Scenario Events

A scenario event represents an incident or condition that exercise participants can respond to, for example a political event, a natural disaster, or some other novel event that might require a participant to respond. VR-Forces does not simulate scenario events. Objects in a simulation do not interact with scenario events and scenario events do not directly affect simulation objects (plans can include conditional statements that are contingent on scenario events). Scenario events are placeholders that allow you to provide a richer simulation experience.

Events can include rich text, audio files (WAV), video (AVI) and raster files (JPG, PNG, SVG, or BMP). Use of these media allow you to expand the scope and quality of a training experience without needing to build a complicated simulation to support it. The flexibility of scenario events allows you to use them in any way that adds value to an exercise, for example, mission briefings, and video teaching points.

Unless you disable it, when an event is triggered, VR-Forces displays a dialog box that displays the event information. If a scenario has events that are triggered by simulation time or time of day, the Next Scheduled Event Toolbar displays the amount of time until the next event (Figure 82).

Figure 82. Next Scheduled Event Toolbar



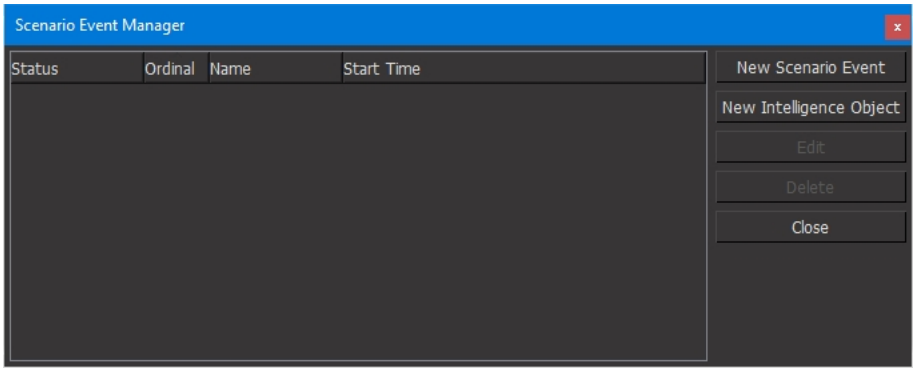
If you want to play MP4 video for scenario events, you must install an appropriate driver on your computer.

8.2. Creating a Scenario Event

➤ **To create a scenario event:**

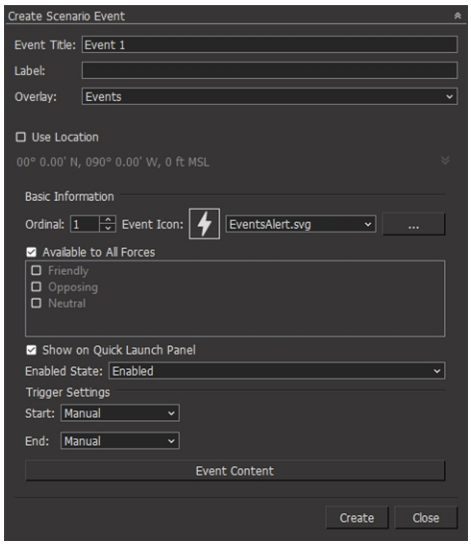
1. Choose **Simulation > Scenario Events** or click the Scenario Events button ( ) in the Scenario Components Toolbar. The Scenario Event Manager opens (Figure 83).

Figure 83. Scenario Event Manager



2. Click New Scenario Event. The Create Scenario Event dialog box opens (Figure 84).

Figure 84. Create Scenario Event dialog box



3. Set values for the parameters in the dialog box. Table 16 describes the parameters you can specify for a scenario event:

Table 16. Scenario event parameters

Parameter	Description
Event Title	A short descriptive name for the event that displays next to the icon on the Quick Launch Panel.
Label	The label provides more information about the event. This is used as the tooltip on the Quick Launch Panel. (optional)
Extended labels	If present, fill out extended labels. (optional)
Overlay	The overlay to which the event is assigned. Default: Events.
Use Location	You can specify a location for an event icon to appear when the event is triggered. You can click on the map where you want the event to occur or enter a location in any of the location formats. (optional)
Ordinal	Events are numbered in the sequence they are created. You can change the numbers to suit your purposes.
Event Icon	Select an appropriate icon to appear on the map at the location and time of the event. This icon is also used on the Quick Launch Panel. (optional)

Table 16. Scenario event parameters (continued)

Parameter	Description
Forces	You can make the event available to all or some forces. To limit it to some of the forces in the scenario, clear the Available to All Forces check box and then select the forces you want to get the event when it occurs. You can select multiple forces.
Show on Quick Launch Panel	Adds the event to the Quick Launch Panel for one-click activation.
Enabled State	Specify whether the scenario event is enabled, disabled, or linked to the enabled state of another scenario event.
Start	Specify one of the options for how the event is triggered: — Manual. Event will be triggered manually by the scenario administrator. — Date/Time. Specify the date and time of day that should trigger the event. — Linked. Specify the linking events as described in "8.2.2 Adding Linked Events" on page 280 that should trigger the event.. — Simulation Time. Specify the simulation time that should trigger the event.. — Immediate. The event starts as soon as it is created.
End	Specify how the end of the event is triggered: — Manual. Ended by the scenario administrator. — Date/Time. Specify the date and time of day at which it should end. — Duration. Specify how long the event lasts. — Permanent. The event does not end until you manually change the end type and end it. ⚠ If you trigger an event by time and the specified time is prior to the start time of the scenario, the event is triggered when the scenario starts.
Event Content	Add content as described in "8.2.1 Adding Content to a Scenario Event" on the facing page.

4. Click Create. Your new event appears in the event list. You can sort events by any of the column headings.

8.2.1 Adding Content to a Scenario Event

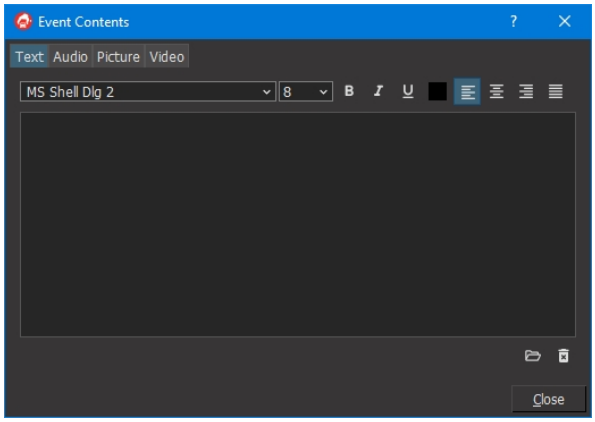
The content for a scenario event can be text, audio, video, graphics (BMP, JPG, SVG, or PNG), or some combination of them.

For sound and video files, you can specify files in any format that your computer supports.

➤ **To add content to a scenario event:**

1. Create an event as described in "8.2. Creating a Scenario Event" on page 276.
2. In the Create Scenario Event dialog box, click Event Content. The Event Contents dialog box opens (Figure 85).

Figure 85. Event Contents dialog box



3. Select the tab for the content type that you want for this event. You can have more than one type of content.
4. For video, picture, or audio:
   a. Click Choose. A file selection dialog box opens.
   b. Select the file you want to add.
   c. Click Open.

   For text:
   a. Type the text you want in the dialog box or click Load and select a text file to load.
   b. Format the text as desired.
5. Optionally, test video and audio files by clicking the Play button.
6. Click Close.

⚠ If the event content is too large, VR-Forces does not save the content when you click Create or Update in the Scenario Event dialog box.

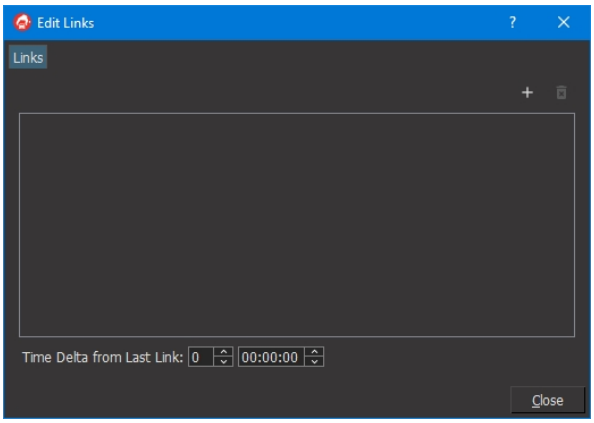
8.2.2 Adding Linked Events

You can link multiple events to a single preceding event (many to one) or link one event to another linked event if your scenario requires it (one to one). You can create a linked event to trigger at any time after the preceding event it is linked to. For example, an event of a bridge being washed away could be linked to a preceding event such as reporting heavy rains, flooding, or rising river levels.

➤ **To add a link to a scenario event:**

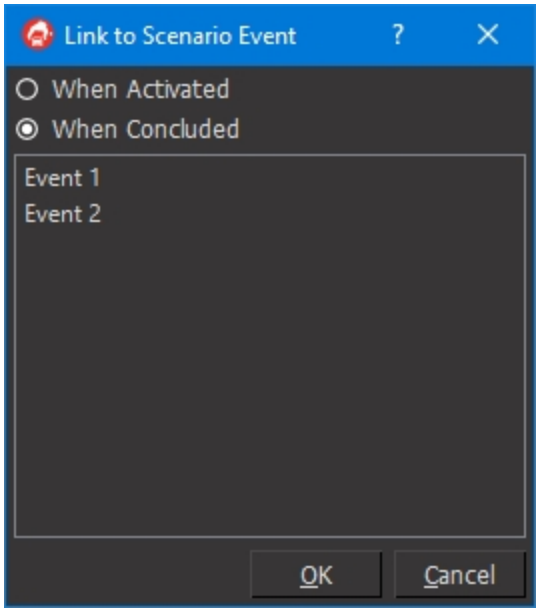
1. Create an event as described in "8.2. Creating a Scenario Event" on page 276.
2. For the Start trigger, select Linked. The Edit Links button is added to the dialog box.
3. Click Edit Links. The Edit Links dialog box opens (Figure 86).

Figure 86. Edit Links dialog box



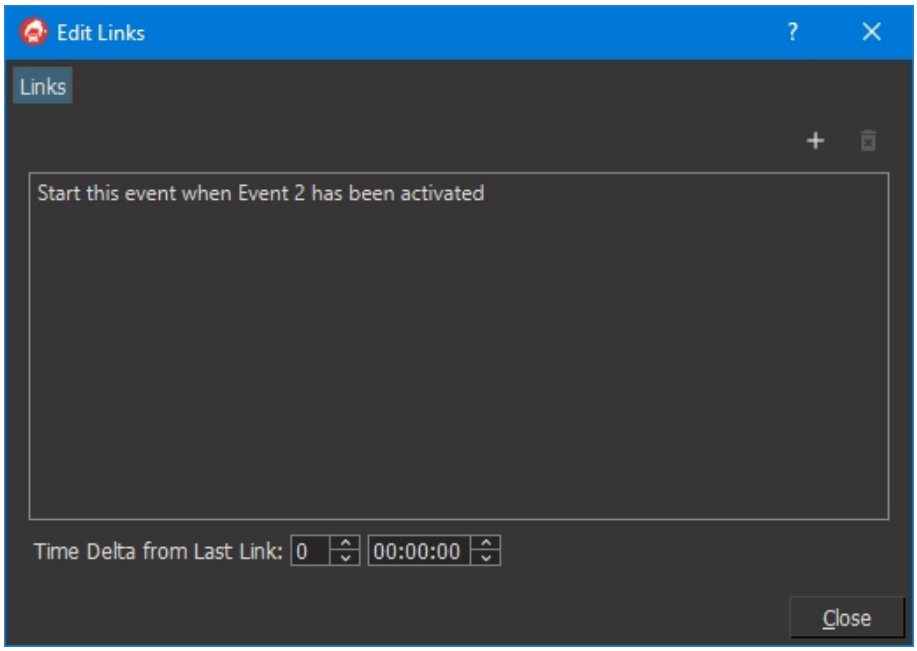
4. Click the Add button (+). The Link to Scenario Event dialog box opens (Figure 87).

Figure 87. Link to Scenario Event dialog box



5. Select When Activated or When Concluded to indicate the relationship of the event to its linked event.
6. Select the event you want to link to.
7. Click OK.
8. Optionally, add additional linked events.
9. Specify the time delta. This is the amount of time to wait after the preceding event before triggering the linked event. If you specify multiple linked events, this is the time after all event conditions have been met.

Figure 88. Edit Links dialog box with links



10. Click OK.

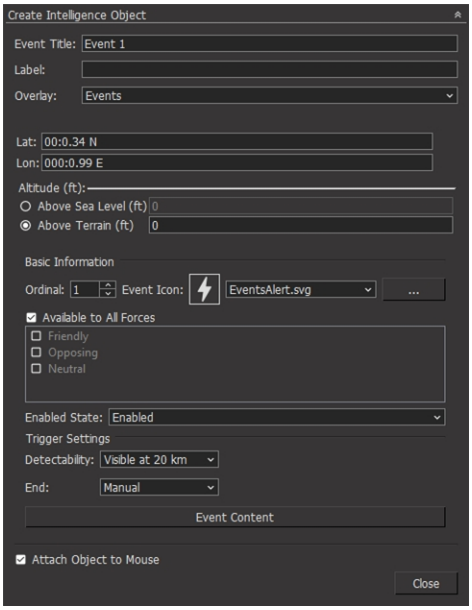
8.3. Adding Intelligence Objects

Intelligence objects are tactical graphics that trigger scenario events when they are detected by a simulation object. By default, intelligence objects are detected by visual sensors. However, you can configure them to be detectable by other sensors.

➤ **To add an intelligence object:**

1. Choose **Simulation > Scenario Events** or click the Scenario Events button ( ) in the Scenario Components Toolbar. The Scenario Event Manager opens (Figure 83 on page 276).
2. Click New Intelligence Object. The cursor changes to draw mode and the Create Intelligence Object dialog box opens (Figure 89).

Figure 89. Create Intelligence Object dialog box



3. Click where you want the intelligence object to be located, or type its location in the dialog box.
4. Specify basic information and trigger settings as you would for a scenario event ("8.2. Creating a Scenario Event" on page 276), except that instead of configuring a start event, you specify detectability.

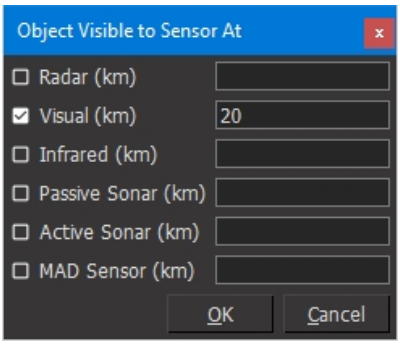
   To use only a visual sensor for detectability, select a distance in the Detectability list.

   To use other sensors, specify detectability:

   a. In the Detectability list, select User Defined. The Edit Detectability button is added to the dialog box.

   b. Click Edit Detectability. The Object Visible to Sensor At dialog box opens (Figure 90).

Figure 90. Set Sensor Signatures



   c. Select each sensor that you want to use to detect the intelligence object.

d. For each sensor specify the sensor signature. (Although sensor signatures are technically unit-less, as a practical matter the sensor signature represents the distance in kilometers at which the object can be sensed.)

e. Click OK.

f. Click Create.

8.4. Starting a Scenario Event

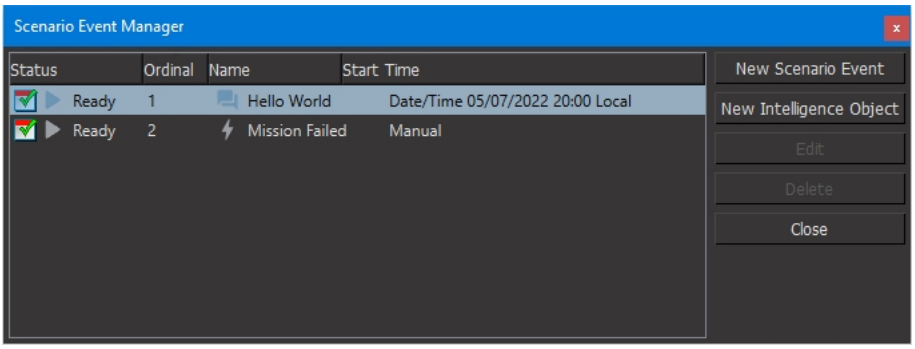
A scenario event starts based on the Start Event trigger, by a command in a plan, or manually. Regardless of how a scenario event's trigger is set up, you can manually start an event at any time.

> If an event has been added to the Quick Launch Panel, you can start it by clicking its icon and title text on the panel.

➤ **To start a scenario event manually:**

1. Choose **Simulation > Scenario Events**. The Scenario Event Manager opens (Figure 91).

Figure 91. Scenario Event Manager, Play button



2. In the Status column, click the Play button (▶) for the event you want to start (Figure 91). The Event window opens (Figure 92).

Figure 92. Event window



8.4.1 Controlling Scenario Events from a Plan

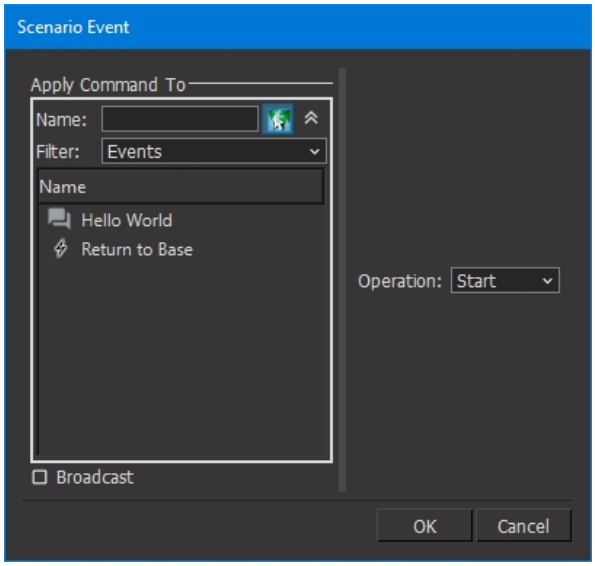
You can start, stop, enable, and disable scenario events from plans. For example, a simulation object's plan could have an Entity In Area condition and when that condition is satisfied, you could trigger an event such as, "The enemy has seized your headquarters!" or "Congratulations, you have seized the enemy headquarters!".

Scenario events can also be used as the condition in a trigger. For details, see "Chapter 42. Plan Concepts" on page 931 and "Chapter 43. Writing Plans" on page 947.

➤ **To send a scenario event in a plan:**

1. In a plan window, choose **Commands > Scenario Event**. The Scenario Event dialog box opens (Figure 93).

Figure 93. Scenario Event dialog box



2. Select the event you want to send.
3. Select an option in the Operation list.
4. Click OK.

8.5. Ending a Scenario Event

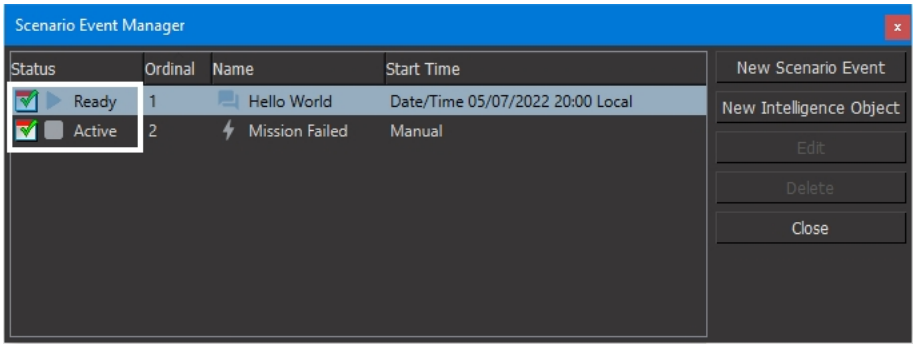
An event ends based on the End Event trigger setting, by a command in a plan, or manually.

When a scenario event ends, the Event window does not close automatically. The title bar indicates that the event is over by changing the text from Active to Inactive. You must close the window manually.

➤ **To end a scenario event manually:**

1. Choose **Simulation > Scenario Events** or click the Scenario Events button ( ) in the Scenario Components Toolbar. The Scenario Event Manager opens.
2. In the Status column, click the Stop ( ) button for the event you want to stop (Figure 94).

Figure 94. Scenario Event Manager, Stop button



If the Status column says Active, but does not have a Stop button, this is a permanent event. To end it, you must edit the event and change the End mode to Manual. A Stop button is then displayed.

3. Click Close.

8.6. Enabling and Disabling Scenario Events

You can enable and disable scenario events. You can change their status from the Scenario Event Manager, the Edit Scenario Event dialog box, and from plans. For details about enabling and disabling a scenario event from a plan, see "8.4.1 Controlling Scenario Events from a Plan" on page 284.

➤ **To enable or disable a scenario event in the Scenario Event Manager:**

1. Choose **Simulation > Scenario Events**. The Scenario Event Manager opens (Figure 94 above).
2. In the Status column, click the Enable/Disable ( ) button for the event you want to stop (Figure 94).

➤ **To enable or disable a scenario event in the Edit Scenario Event dialog box:**

1. Choose **Simulation > Scenario Events**. The Scenario Event Manager opens (Figure 94).
2. Select the event you want to edit.
3. Click Edit. The Edit Scenario Event dialog box opens (similar to Figure 84 on page 277).
4. In the Enabled State list, select the option you want.
5. Click Update.

You can also click the Scenario Events button ( ) in the Scenario Components Toolbar to open the Scenario Event Manager.

8.7. Sorting the Event List

Scenario events have ID numbers. The numbers have no effect on when events are sent. They are simply a convenience for instructors and players. You can sort the list by number or by any other column listed.

➤ To change the sort order of scenario events, click the heading of the column that you want to sort on.

8.8. Deleting a Scenario Event

➤ **To delete a scenario event:**

1. Choose **Simulation > Scenario Events** or click the Scenario Events button ( ) in the Scenario Components Toolbar. The Scenario Event Manager opens (Figure 83 on page 276).
2. Select the event you want to delete.
3. Click Delete.
4. Click Yes in the confirmation message.

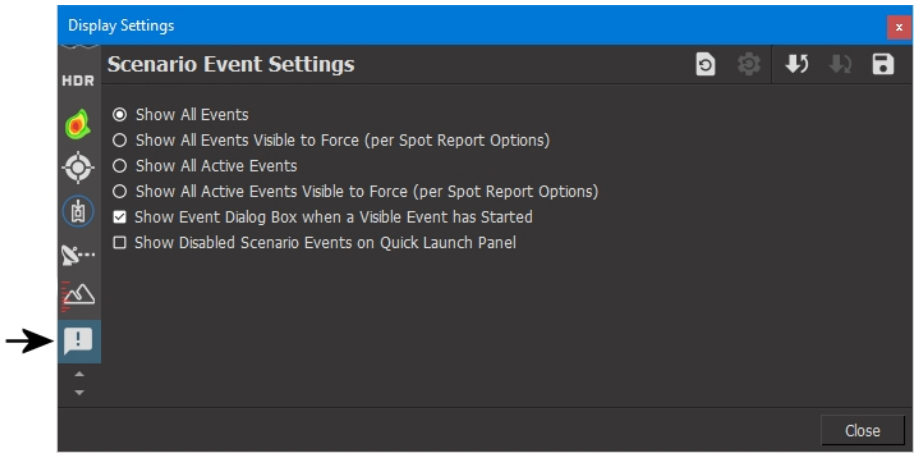
8.9. Showing and Hiding Scenario Events

You can specify which events, if any, display and whether the Event dialog box displays when an event is triggered. The visible to force settings refer to visibility according to spot report settings and are based on the force availability specified when a scenario event is created or edited.

➤ **To show or hide scenario events:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Scenario Event Settings page (Figure 95).

Figure 95. Scenario Event Settings page



3. Select an option for viewing events.



9. Target Detection and Combat Features

This chapter describes VR-Forces features related to target detection and acquisition, such as spot reports, hostility relationships, lasing, and munition damage.

9.1. Displaying Spot-Reported Simulation Objects 290
9.1.1 Enabling or Disabling Spot Reports 290
9.1.2 Configuring the Spot Reports Viewpoint 291
9.1.3 Configuring the Spot Reports Certainty Level 294
9.1.4 Applying Spot Reports to Tactical Graphics 295
9.1.5 Displaying Entity Ground Truth in 3D Observer View 295
9.1.6 Displaying Labels for Spot Reports 296
9.1.7 Using Spot Reports in Tasks 296
9.1.8 Displaying Track Histories for Spot-Reported Objects 296
9.2. Displaying Tactical Data Link Messages 298
9.2.1 Configuring the Display of TDL Messages 299
9.2.2 Displaying Track Histories for TDL Messages 300
9.3. Managing Force Hostility Relationships 300
9.3.1 Changing a Force's Hostility in a Plan 302
9.4. Detecting Targets 302
9.4.1 Target Detection and Spot Reports 304
9.4.2 Lasing Targets 304
9.5. Engagement Zones 305
9.6. Electronic Warfare 306
9.7. Radar Coverage 306
9.7.1 Enabling the Display of Radar Coverage 308
9.7.2 Displaying Radar Coverage 308
9.7.3 Configuring Radar Coverage Displays 309
9.8. Using Sonar 310
9.8.1 Propulsion Noise and Sonar 310
9.9. Launching Countermeasures (Chaff and Flare) 311

9.10. Modeling Artillery Munitions 312

9.1. Displaying Spot-Reported Simulation Objects

A spot report is a report by a simulation object to members of its force, notifying them that it has detected a member of another force.

By default, VR-Forces displays simulation objects based on complete ground truth. That is, it shows all simulation objects in their exact location. In real-world combat and training situations, however, participants rarely know where all other participants are, including those in their own force. This is called the fog of war. VR-Forces can simulate the fog of war by displaying simulation objects based on spot reports rather than ground truth.

When you display simulation objects based on spot reports, you do not see all members of forces other than your own. You only see spot-report icons, also known as tracks, for other-force simulation objects that have been detected by members of your force. Spot-report icons are similar to simulation object icons, but they do not move as simulation object icons do. Furthermore, since spot reports lose value over time as the detected entities may move to different locations, VR-Forces maintains a certainty level for the location of spot-reported entities. If a detected entity is not detected again for a period of time, its certainty level decreases and its icon becomes increasingly transparent to show that it is less certain that the entity is at the reported location.

- Generating spot reports can reduce the performance of the simulation engine.
- When spot reports are enabled, VR-Forces does not display icons for dead entities.
- When you rewind a scenario, the global spot report setting persists. Spot report settings for individual entities revert to the settings saved with the scenario.
- If you enable spot reports and you use Stealth observer mode, VR-Forces uses 2D icons for the entities that are identified by spot reports. You do not see any 3D models.

You can find links to videos that demonstrate simulation object classification and spot reports at https://www.mak.com/mak-one/support/help/tutorials, or click the camera icon next to the procedure whose video you want to view.

9.1.1 Enabling or Disabling Spot Reports

You can enable or disable spot reports globally or per simulation object. To set spot reports for a simulation object, use the Spot Reports set data request, as described in "40.77. Spot Reports" on page 901.

By default, when you enable spot reports, they are sent only to the GUI (for display purposes). They are not sent to other simulation objects. This reduces the performance impact of spot reports on the sim engine, but it means that entities do not receive that information for targeting purposes. Their spot report receivers do not show any spot reports in the Information dialog box. If you want spot reports to be sent to simulation objects, you must enable spot reports as described here and also specify that spot reports be sent to simulation objects using the Spot Report set data request. For more information, see "40.77. Spot Reports" on page 901.

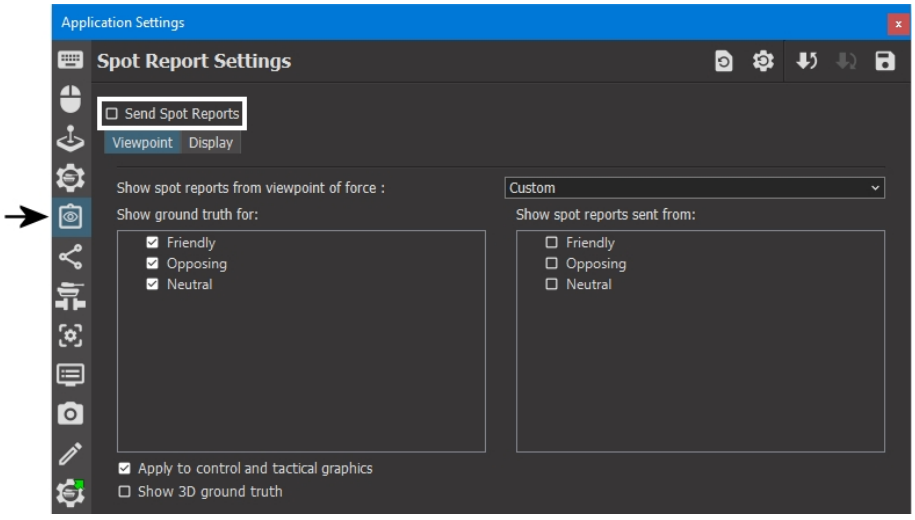
Enabling or Disabling Spot Reports Globally

When you enable or disable spot reports globally, the setting is applied to all simulation objects that have not explicitly set spot reports on or off with the Spot Reports set data request. The setting persists, which allows you to set a simulation object to use the global spot report setting.

➤ **To enable or disable spot reports for all simulation objects:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 96).

Figure 96. Spot Report Settings page, Viewpoint tab



4. Select or clear the Send Spot Reports check box. The change takes effect immediately.

9.1.2 Configuring the Spot Reports Viewpoint

The spot report viewpoint determines which simulation objects and tactical graphics are shown using ground truth and which are shown using only spot-report icons. You can select preconfigured viewpoints or create custom viewpoints.

- If you create a simulation object for a force that is not visible based on the viewpoint setting, after you create the simulation object, it disappears. Therefore, we recommend that you show ground truth for all forces when you are creating entities, tactical graphics, simulation effect objects, and so on.
- When you rewind a scenario, the viewpoint configuration stays the same as in the previous run of the scenario. The global spot report setting stays the same. Entity-specific spot report settings revert to the settings saved with the scenario.

Selecting a Preconfigured Spot Report Viewpoint

VR-Forces provides viewpoints for the friendly, opposing, and neutral forces. In each of these viewpoints, VR-Forces displays ground truth for the viewpoint force and spot-report icons for the other forces. For example, if you select the Friendly force viewpoint, all simulation objects in Force 1 are displayed showing ground truth. All other simulation objects are displayed only if they have been detected by a friendly simulation object that is configured to send spot reports.

When you select a preconfigured viewpoint, the check boxes in the windows in the Map View group box are automatically set to match the selected viewpoint.

➤ **To configure spot reports:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 96 on the previous page).
4. Select an option in the Show Spot Reports from Viewpoint of Force list.

Configuring Custom Spot Report Viewpoints

If you do not want to use a preconfigured viewpoint, you can create a custom viewpoint. This option allows you to see ground truth and spot-report icons from a mix of forces. When you create a custom viewpoint, the Show Spot Reports from Viewpoint of Force list displays the viewpoint that you have created (usually Custom).

It is possible to create a custom configuration in which no objects are displayed, so think carefully about your choices.

➤ **To configure a custom viewpoint:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 96 on the previous page).
4. In the Show Ground Truth For window, select the forces for whom you want to display the ground truth. You would usually select the same forces as the ones selected in the Show Spot Reports Sent From window.
5. In the Show Spot Reports Sent From window, select the forces whose spot-report icons you want to display. In other words, if you are not showing ground truth for the opposing force and only want to see the opposing force simulation objects that

are detected by friendly and neutral simulation objects, then select the Friendly and Neutral check boxes.

Configuring Same-Side Fog of War

There may be cases when you want to simulate same-side fog of war. That is, members of a force do not know where other members of their force are unless detected by a simulation object and reported using a spot report.

To simulate same-side fog of war, you must edit the *spot-report-generator.sysdef* file. You might want to create a customized file that you can assign only to the simulation objects that you want to simulate same-side fog of war. (You can use the Simulation Object Editor to change the spot report systems that simulation objects use.) In the system definition file change the **send-spot-reports-on-own-force** parameter to True, as shown on line 14:

```
1   (spot-report-generator-system
2     (controllers
3       (spot-report-generator
4         (component-descriptor-type
5           "spot-report-generator-controller-descriptor")
6         ...
7         (send-stop-spotting-reports False)
8         (object-types-to-spot-report
9           (object-type 1 (1 -1 -1 -1 -1 -1 -1))
10          (object-type 1 (2 -1 -1 -1 -1 -1 -1))
11          (object-type 1 (3 -1 -1 -1 -1 -1 -1))
12          (object-type 1 (5 -1 -1 -1 -1 -1 -1))
13        )
14        (send-spot-reports-on-own-force True)
15        (send-spot-reports-on-hostile-forces True)
16        (send-spot-reports-on-neutral-forces True)
17        (broadcast False)
18        ...
19        (send-spot-reports-on-scenario-load True)
20        (send-to-front-end-only True)
21        (only-one-destroyed-report False)
22        (send-only-on-data-change True)
23      )
24    )
25    ...
26  )
```

Configuring Other Options for Spot Reports

You can change other defaults in the sysdef file, including limiting the **object-types-to-spot-report**. Reducing the number of objects detected makes spot reporting less computationally expensive. Another parameter that you might want to change to reduce the number of objects is **send-spot-reports-on-neutral-forces** by setting it to False.

You can also change the values of the **send-only-to-front-end** and **broadcast** parameters to change the default behavior of spot reports and how they affect performance. Setting **send-only-to-front-end** to False sends spot reports to other sim engines and simulation objects. Setting broadcast to true sends spot reports to all simulation objects in the same force.

The **send-stop-spotting-reports** parameter controls when to send out stop spotting reports. When it is set to false, VR-Forces sends stop spotting reports only when an entity is acting as a remote sensor. Entities receiving spot reports from a remote sensor treat them as if they had detected the spotted objects themselves. When the parameter is set to true, they are sent whenever detection is lost, not only for remote sensor receivers. You do not need to wait for a timeout to know that an entity is no longer detected.

9.1.3 Configuring the Spot Reports Certainty Level

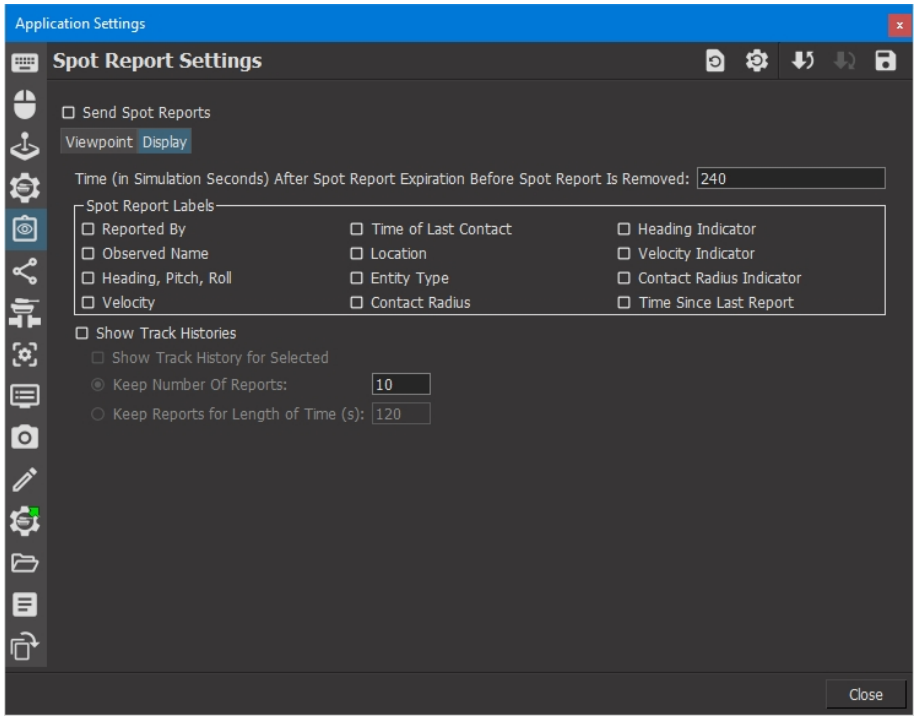
It is assumed that as time passes the validity, or certainty, of a spot report declines. To simulate this decline in certainty, VR-Forces can display the icons that represent spot-reported entities at increasing levels of transparency. Once the initial validity period has expired, the spot report degrades over a configurable amount of time. The icon's transparency decreases in four steps, with each step being 25% of the total time allotted for degradation.

> The validity period is specified by the **report-resend-period** parameter in the *spot-report-generator.sysdef* file in the SMS you are using. The default is 60 seconds.

➤ **To configure the amount of time, in seconds, over which a spot report degrades:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 97).
4. Specify a value, in seconds, in the Time After Spot Report Expiration Before Spot Report Is Removed box.

Figure 97. Spot Reports Option, Display tab



9.1.4 Applying Spot Reports to Tactical Graphics

If a tactical graphic is assigned to a force, you can display it based on spot reports rather than ground truth. The same criteria are applied to their display as are applied to the display of simulation objects. For information about how to assign objects to a force, see "47.1. Creating Tactical Graphics" on page 1012.

➤ **To apply spot reports to tactical graphics:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 96 on page 291).
4. Select the Apply to Control and Tactical Graphics check box.

9.1.5 Displaying Entity Ground Truth in 3D Observer View

When you display simulation objects based on spot reports, the default is to hide entities in any force except your own in all observer views. However, you can choose to display them in the 3D (Stealth) view regardless of their combat identification level (CID). This adds more realism to fog-of-war scenarios, where a person might be able to see something when it has not yet been detected by an simulation object. Note that you cannot interact with simulation objects that are considered hidden by fog of war, even when you can see it in the 3D view.

➤ **To specify whether entity ground truth displays in the 3D when spot reports are enabled:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Viewpoint tab (Figure 96 on page 291).
4. Select or clear the Show 3D Ground Truth check box.

9.1.6 Displaying Labels for Spot Reports

You can display labels (symbol decorations) for the icons of spot-reported entities with information similar to that for regular simulation object icons.

➤ **To display spot report labels:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 97 on the previous page).
4. In the Spot Report Labels group box, select the labels that you want to display.

ⓘ You can also enable labels for spot-reported entity icons on the Display Settings dialog box, Symbol Decoration Settings page.

9.1.7 Using Spot Reports in Tasks

Spot-reported simulation objects are listed in the Objects List Panel and other simulation object lists. Although the spot-report icon in the display window may not show information about a spot-reported object if it has not been identified, the entry in the list includes the object name. You can filter the list to show only spot-report icons.

You can select a spot-reported object as a parameter in tasks that require you to select a simulation object. If you use a spot-reported simulation object in a task, VR-Forces carries out the task based on the ground truth of the reported simulation object, not the location of the spot-report icon. For example, if you specify a spot-reported object in the Follow Entity task, and the target simulation object moves, the following simulation object will move toward the actual location of the spot-reported simulation object, not its location when the spot report was made.

9.1.8 Displaying Track Histories for Spot-Reported Objects

You can display track histories for spot-reported simulation objects (Figure 98). When you enable display of track histories for spot-reported objects, it does not actually cause them to display. You must also enable them for the selected spot-reported object and select a spot report.

Figure 98. Spot report track history



The Track Histories setting on the Observer menu and the Observer toolbar do not affect the display of track histories for spot-reported objects. That feature shows entities' histories.

➤ **To display track histories for a spot-reported simulation object:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Spot Report Settings page.
3. Select the Display tab (Figure 97 on page 295).
4. Select the Show Track Histories check box.
5. Select the Show Track History for Selected check box.
6. Select the spot-reported object whose track history you want to show.

Pinning a Spot Report Track History

If you want to see more than one spot report track history at a time, you can pin track histories. If a track history is pinned, you do not have to select the spot-reported simulation object to see its track history.

➤ **To pin the track history for a spot-reported object:**

1. Select the spot-reported simulation object.
2. Choose **Objects > Decorations > Pin Track History**.

Removing Spot Report Track History Pins

➤ **To remove spot report track history pin:**

1. Select a simulation object that has its track history pinned.
2. Choose **Objects > Decorations > Pin Track History**. The track history pin is removed.

➤ To remove track history pins for all entities, choose **Objects > Decorations > Remove All Pin Track History**.

9.2. Displaying Tactical Data Link Messages

VR-Forces can display information about tactical data link (TDL) messages received over the network or sent by simulation objects that have a tactical data link message generator system. Systems are provided for each major platform type. TDL support is provided by an ITAR-controlled plug-in. However, if you do not have the plug-in, the base VR-Forces product simulates the effect of generating TDL messages.

When VR-Forces receives a TDL message, it displays:

- An icon similar to a spot report icon.
- Optionally, the time since the last report.
- Optionally, the data that accompanies the message, in up to 9 lines. The number of lines to display is configurable. There is no particular type of information or ordering information associated with the lines. It is simply based on the data in the TDL message.
- Optionally, track histories for the TDL reports.

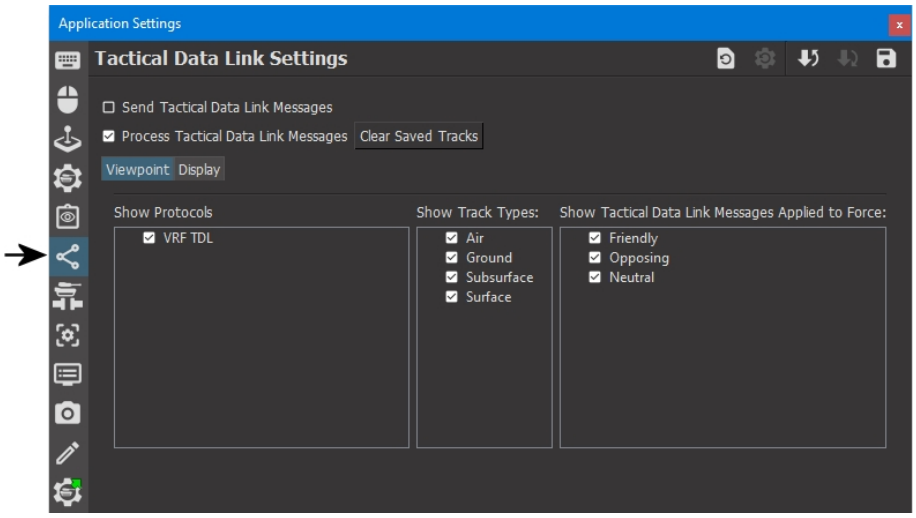
For information about configuring the display of TDL messages, see "9.2.1 Configuring the Display of TDL Messages" on the facing page.

You can enable TDL messaging globally or for individual simulation objects. To enable TDL messages for a specific simulation object, use the TDL Messaging set data request. For details, see "41.49. TDL Messaging" on page 927.

➤ **To enable TDL messaging globally:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page (Figure 99).

Figure 99. Tactical Data Link Settings page, Viewpoint tab



3. To send TDL messages, select the Send Tactical Data Link Messages check box.
4. To display TDL messages, select the Process Tactical Data Link Messages.

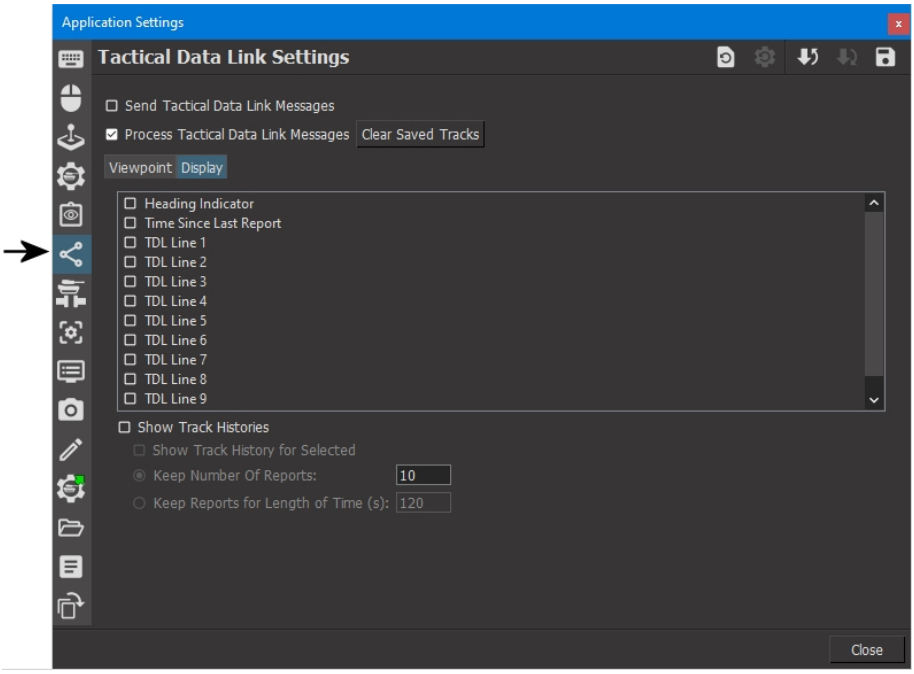
9.2.1 Configuring the Display of TDL Messages

You can configure how VR-Forces displays TDL messages.

➤ **To configure TDL messaging:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page.
3. Select the Viewpoint tab (Figure 96 on page 291).
4. In the Show Protocols list, select the type of protocol for which you want to show messages. (If the ITAR plug-in is installed, the VR-Forces TDL messages are not sent.)
5. In the Show Track Types list, select the platforms for which you want to show messages.
6. In the force list, select the forces from which you want to display messages.
7. Select the Display tab (Figure 100).

Figure 100. Tactical Data Link Settings, Display page



8. Select check boxes for the data you want displayed next to the TDL message icon.
9. Optionally, select the Show Track Histories check box.
10. If you enable track histories, specify whether to show track histories only for the selected entity, the number of reports to track, and how long to display them. For

information about how to display track histories, see "9.2.2 Displaying Track Histories for TDL Messages" below.

9.2.2 Displaying Track Histories for TDL Messages

You can display track histories for tactical data link (TDL) messages. When you enable display of track histories for TDL messages, it does not actually cause them to be displayed. You must also enable them for the selected message and select a message.

> The Track Histories setting on the Observer menu and the Observer toolbar do not affect the display of track histories for TDL messages.

➤ **To display track histories for a TDL message:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Tactical Data Link Settings page.
3. Select the Display tab (Figure 100 on the previous page).
4. Select the Show Track Histories check box.
5. Select the Show Track History for Selected check box.
6. Select the TDL message whose track history you want to show.

Pinning a TDL Message Track History

If you want to see more than one TDL message track history at a time, you can pin track histories. If a track history is pinned, you do not have to select the message to see its track history.

➤ **To pin the track history for a TDL message:**

1. Select the message.
2. Choose **Objects > Decorations > Pin Track History**.

Removing TDL Message Track History Pins

➤ **To remove TDL message track history pins:**

1. Select a simulation object that has its track history pinned.
2. Choose **Objects > Decorations > Pin Track History**. The track history pin is removed.

➤ To remove track history pins for all entities, choose **Objects > Decorations > Remove All Pin Track History**.

9.3. Managing Force Hostility Relationships

VR-Forces categorizes simulation objects as members of a force, such as friendly, opposing, or neutral. By default, friendly forces and opposing forces are hostile to each other. Neither is hostile towards neutral forces. You can change force hostility

relationships dynamically. This may be particularly useful if you are simulating an environment in which civilian populations become hostile to a particular force, or in which armed militias, for example, switch allegiances to other forces.

You can add new forces in the Simulation Object Editor. For details, see "68.12. Managing the Forces List" on page 1331. You can change a force's hostility in a global or individual plan. For details, see "9.3.1 Changing a Force's Hostility in a Plan" on the next page.

You can change the default hostility relationships by editing the *forceHostilty.mtl* file for the simulation model set you are using (for example, *./data/simulationModelSets/<model_set>/forceHostilty.mtl*).

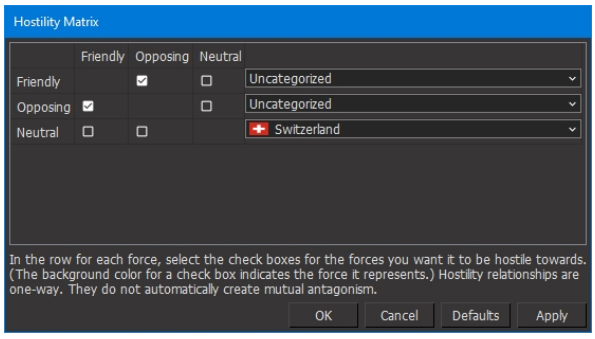
When you change force hostility at runtime, the changes are saved as part of the scenario. The hostility relationships loaded with a scenario override the hostility relationships loaded by the simulation model set's hostility file.

- Hostility relationships are not reciprocal. For example, if you set neutral force simulation objects to be hostile to friendly force simulation objects, that does not mean that friendly force simulation objects are hostile to neutral force simulation objects.
- If you click the Defaults button, it resets the Hostility Matrix to the values when you first opened the dialog box in the VR-Forces GUI during the current session. If the you have not made any changes, clicking the Defaults button has no effect.

➤ **To change the force hostility matrix:**

1. Choose **Simulation > Hostility Matrix**. The Hostility Matrix dialog box opens (Figure 101). Each row in the matrix lists the hostility relationships between that row's force and all of the other forces (represented by color-coded columns).

Figure 101. Hostility Matrix



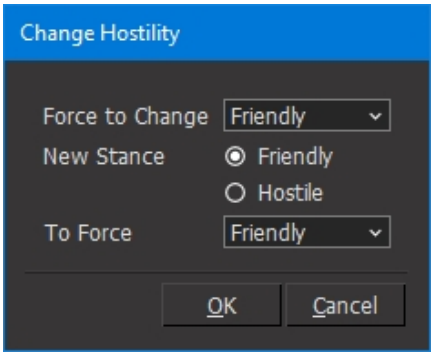
2. For each row in the matrix, select the check boxes in the columns for the forces that you want this force to be hostile towards. Clear the check boxes for forces that you do not want it to be hostile towards.
3. If desired, assign a country to a force by choosing it from the drop-down. When you create an entity for that force and the entity has markings defined for that country, then the entity will automatically display those markings.
4. Click OK.

9.3.1 Changing a Force's Hostility in a Plan

➤ **To change a force's hostility in a global or local plan:**

1. Open a global plan or an individual plan.
2. Choose **Commands > Change Hostility**. The Change Hostility dialog box opens (Figure 102).

Figure 102. Change Hostility dialog box



3. In the Force to Change list, select the force whose hostility you want to change.
4. Select the new force relationship (New Stance), either Friendly or Hostile.
5. In the To Force list, select the force towards which you want to change hostility.
6. Click OK. The command is added to the plan.

9.4. Detecting Targets

Simulation objects can have one or more sensors, for example, visual, sonar, or radar, for detecting and identifying other simulation objects. If a simulation object has a Remote Sensor Receiver system, it can use another entity's sensors to discover sensor contacts. (For more information about remote sensors, see "40.64. Remote Sensor" on page 895.)



When a simulation object detects another simulation object, it assigns it a combat identification (CID) level. If it continues to maintain contact with the simulation object, it upgrades the CID level:

- CID Level 0 – not detected.
- CID Level 1 – Detected: the simulation object is detected, but only its platform is known, for example, ground.
- CID Level 2 – Classified: the simulation object's category is known, for example, tank.
- CID Level 3 – Identified: the simulation object's category and subcategory are known, for example M1A2.
- CID Level 4 – Full Knowledge: all visible information about the simulation object is known.

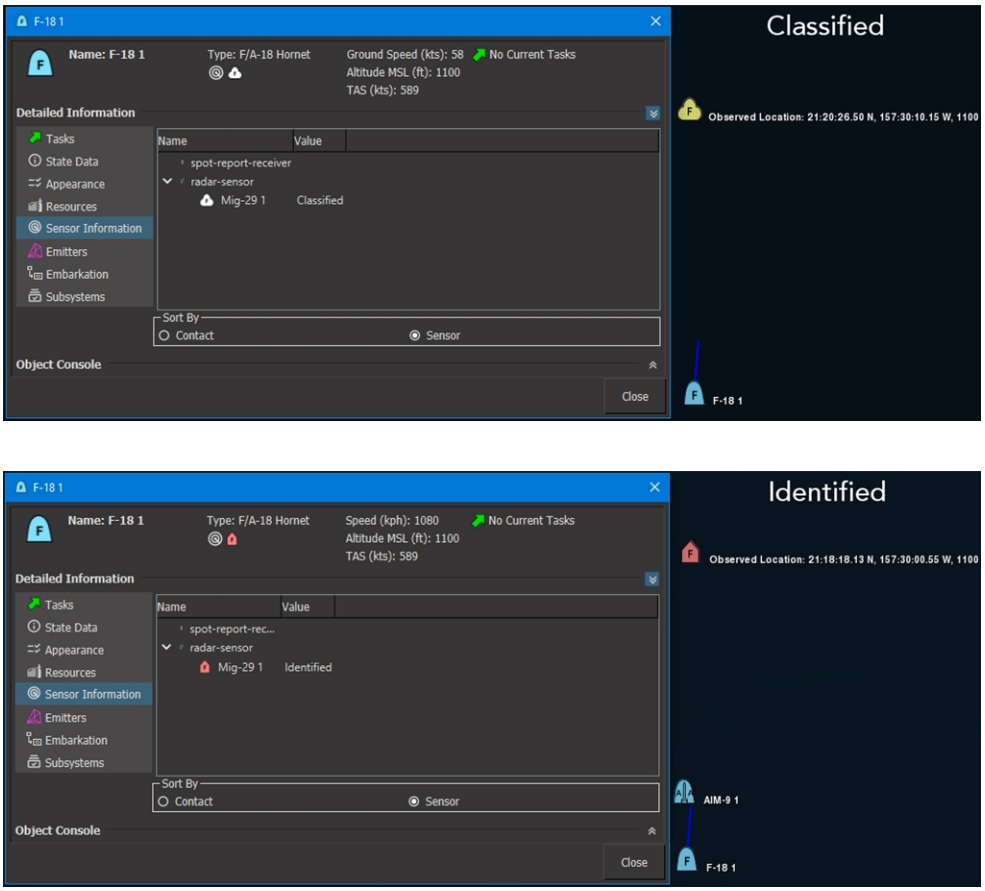
CID levels are assigned based on the distance to the detected simulation object and the length of time that the simulation object has been in contact. You can configure these values in the detection tables (*./data/simulationModelSets/<model_set>/vrfSim/detection/*.csv*). For details about these configuration files, see "76.2. Detection Tables" on page 1516.

By default, simulation objects do not fire at a detected hostile simulation object until it achieves the Identified level. You can configure this behavior by changing the **detection-level-to-set-hostility** parameter in the simulation object's sensor system.

Figure 103 shows screen captures of a simulation object as its CID level is advanced from Classified through Identified. In this scenario, spot reports are enabled. Therefore, when a simulation object is detected, it displays with a yellow icon, rather than the normal color for the force. It is not given the correct force color until the detected simulation object is CID level Identified.

You can find links to videos that demonstrate simulation object classification and spot reports at https://www.mak.com/mak-one/support/help/tutorials, or click the camera icon next to the procedure whose video you want to view.

Figure 103. Simulation object classification and display when spot reports are enabled



9.4.1 Target Detection and Spot Reports

Simulation object detection and classification can be enhanced by enabling spot reports. Detected simulation objects are assigned CID levels based on length of contact and distance from the detecting simulation object. In some cases, such as a fixed-wing aircraft flying quickly over a battlefield, or a simulation object that is at the farthest point of its range from a target, contact might not be sufficient to advance to the Identified CID level. In such cases, even though the simulation object is technically capable of firing on the target, it might not do so because the CID level is not high enough.

If spot reports are enabled, the classification of simulation objects in the spot reports that a simulation object receives is combined with the directly sensed information. Therefore, if a simulation object classifies a target at a low CID level, but it receives a spot report that classifies the target at a high CID level, it can fire on the target based on the CID level in the spot report. For example, as noted in the previous paragraph, a fixed-wing aircraft is unlikely to maintain contact with a ground-based simulation object long enough to elevate its CID level to Identified. However, if a member of its force on the ground identifies the target and sends out a spot report indicating CID level 3 or 4, when the fixed-wing aircraft receives the spot report, it can add the target to its target list.

9.4.2  Lasing Targets

Some entities, such as DI Lasing, have laser-targeting capability. Others, such as attack helicopters, have laser targeting capability and carry laser-guided missiles. Laser-guided missiles use the standard VR-Forces targeting mechanisms. They fire on targets that are identified by laser beams that have the same laser code as the laser-guided missile launcher. The laser beam could be generated by a laser designator on the same entity (autonomous lasing), or could be generated by another entity. Entities with both a laser designator and a missile launcher use their laser and missile when autonomously engaging targets.

Laser codes are numbers in the range of 1111-1788 using the digits 1 through 8. Three-digit codes in the range 111-788 are also supported with an implied initial 1 in the code. By default, VR-Forces calculates a unique laser code for each entity when it is created. When you assign a laser code to an entity, any laser beams sent by that entity use the assigned laser code and the missiles carried by that entity (if applicable) use the same code. Therefore, for example, the default behavior for attack helicopters would be to fire their laser-targeted missiles only at targets that they have lased.

> A laser-guided missile launcher must be able to see a laser spot and an appropriate target before it fires. If the missile loses the laser spot before it hits the target, it tracks to the last known laser spot.

Autonomous Lasing

By default, entities that have laser-guided missiles are configured for autonomous lasing. This means that they can lase targets and fire at them without outside intervention (for example, using the Lase Target task). Entities with autonomous lasing enabled identify and fire at targets independently, just like entities that use conventional munitions.

Synchronizing Laser Codes

There are cases in which you will want more than one entity to use the same laser code. For example, you might want one entity to lase targets (the designator), and another one to fire on them (the shooter). In this case you would assign both entities the same laser code. The designator would be given the hold-fire rule of engagement or not have any missile resources. The shooter would be configured to not lase targets (Lase Autonomous would be set to Off). It would just fire on the targets identified by the designator's laser.

You can synchronize the laser codes of the lasing entity and the firing entity by setting them for each entity with the Laser Code set data request, or you can let VR-Forces synchronize the codes with the Synchronize Laser Code set data request. For details, see "40.46. Laser Code" on page 885 and "40.83. Synchronize Laser Code" on page 903.

9.5. Engagement Zones

Engagement zones are areas associated with ground-based simulation objects and air units that affect their rules of engagement. VR-Forces supports two types of engagement zones: missile engagement zones (MEZ) and joint engagement zones (JEZ). Engagement zones can have infinite height, like most VR-Forces areas, or can have a floor and ceiling, like extruded areas.

MEZs usually refer to zones associated with a ground-based air defense system. The system is permitted to engage targets within this defined area. Multiple engagement zones can be associated with a single air defense system and multiple systems can be associated with an engagement zone. To associate simulation objects with an engagement zone, you use the Engagement Zone set data request. For details, see "40.31. Engagement Zones" on page 879.

JEZs are zones in which multiple systems can engage targets, usually both air units and ground based units. Successful management of fire in a JEZ may require a mechanism to manage the various systems and to help prevent friendly fire.

When a simulation object that is associated with an engagement zone detects a target, VR-Forces checks its rules of engagement. If the ROE is set to Fire When in Engagement Zone and if the target is in the engagement zone, it can fire on the target. If the target is not in the engagement zone, it cannot fire at it.

Ordinarily, simulation objects do not fire on other simulation objects unless they have a combat identification level of Identified or Full Knowledge. An engagement zone can be set to assume that unknown objects are hostile. In this case, if a simulation object has the Fire When in Engagement Zone rule of engagement, it can fire on unknown objects. For more information, see "40.11. Assume Unknown is Hostile" on page 870.

Radars detect the position and type of potential targets, but not the force. Therefore, when a simulation object is set to fire on unknown objects, it will fire on objects in its own force if they do not identify themselves as being of the same force. The force of a target is known only if it has an active transponder (IFF is enabled) and it is of the same force as the radar.

Like other areas, engagement zones can be activated and deactivated. If an engagement zone is inactive, simulation objects whose ROE is set to Fire When in Engagement Zone cannot fire at targets when they are in the zone.

You can create engagement zones just like you would other areal objects. For details, see "Chapter 14. Creating and Placing Objects" on page 371 and "47.1.8 Creating Volumetric (Extruded) Tactical Graphics" on page 1016.

9.6.  Electronic Warfare

In entity-level scenarios, VR-Forces supports radar jamming by entities that have a tactical radar jammer system. The jamming entity must have line-of-sight to the jammed entity and the jammed entity must be within the cone of the electromagnetic emissions from the jamming entity. (For information about electromagnetic emissions, see "50.12. Displaying Electromagnetic Emissions" on page 1072.)

Radar jamming is enabled and set up using the set data requests:

- "40.2. Add Jamming Target" on page 866.
- "40.44. Jammer State" on page 885.
- "40.45. Jamming Mode" on page 885.
- "40.65. Remove Jamming Target" on page 895.

There is also a model for jamming at the aggregate level. See "27.4. Electronic Warfare" on page 539.

9.7. Radar Coverage

VR-Forces can display the area in which a simulation object's radar can detect objects. The radar coverage area is color coded based on the altitude at which it is testing intersections. Each color shown indicates full visibility at that altitude and above. A sensor might be able to detect simulation objects at a lower altitude, but this is not certain. You can specify the altitudes at which you want to test intersections and can specify the colors used to display the radar coverage.

In 2D, the coverage area is depicted as bands of color (Figure 104). The color indicates the minimum altitude layer at which the radar can see in that direction.

Figure 104. Radar coverage, 2D and color key

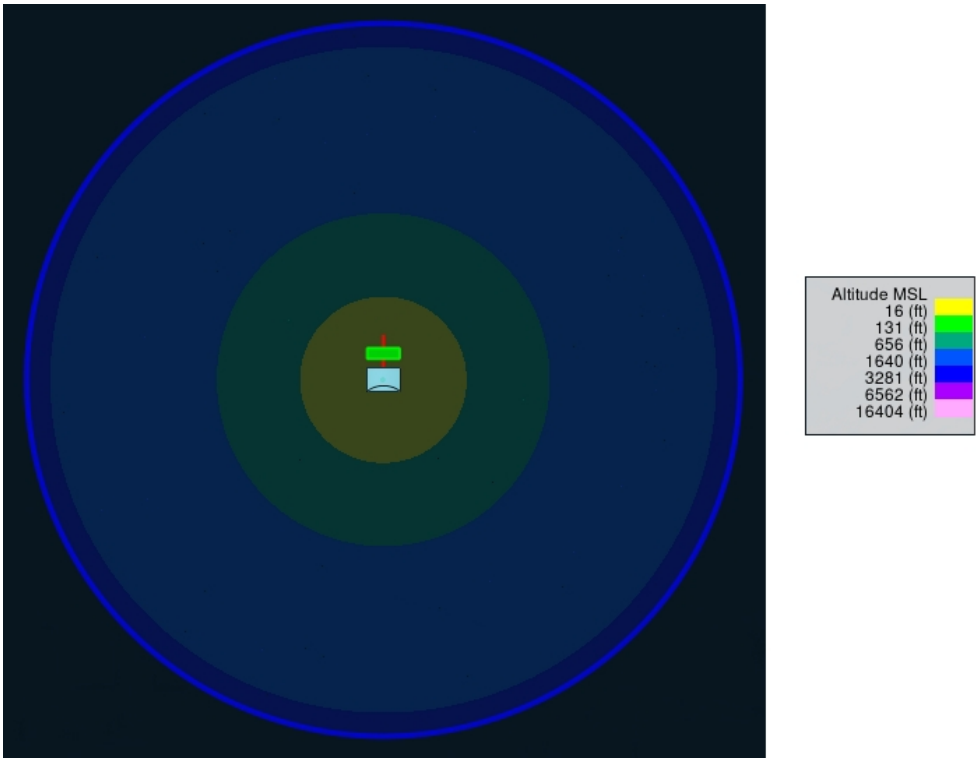
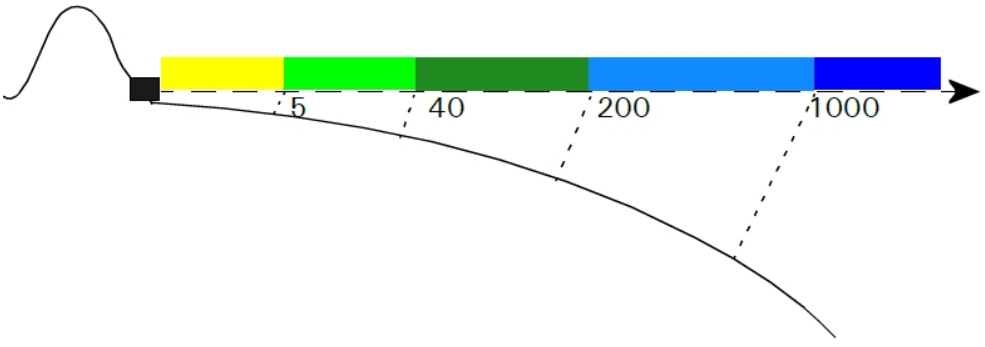


Figure 105 illustrates how the radar coverage display is calculated. Imagine a radar facility at sea level looking out over the ocean (similar to Figure 104). The dotted lines represent the height above sea level of the radar beam as the earth curves (the illustration is not to scale). In the yellow band, the radar can detect all objects that are 5 meters above sea level or higher. In the light green band, the radar can detect all objects that are 40 m above sea level or higher. It might also detect objects between 5 m and 40 m. For each specified intersection height that you want to test, VR-Forces adds another band of color.

Figure 105. Radar coverage



In 3D, the coverage area is depicted as a shaded section of a sphere surrounding the radar. Areas covered by the radar are shaded. Areas that are obscured have no shading. A color gradient is used to show varying altitudes.

You can configure altitude layers and colors on the Display Settings dialog box, Radar Coverage Settings page.

- Radar coverage displays can take some time to calculate, and fill in gradually as the calculation proceeds.
- Because the coverage area is computationally expensive to display, this feature is only usable on units and entities that are stationary.
- Radar coverage ignores vegetation (procedural trees) and structures (buildings and SpeedTrees).

9.7.1 Enabling the Display of Radar Coverage

Radar coverage is enabled or disabled globally per observer. It is displayed per simulation object. For details about displaying radar coverage, see "9.7.2 Displaying Radar Coverage" below.

➤ **To enable the display of radar coverage, use one of the following methods:**

- Choose **Observer > Radar Coverage**.
- On the Observer Toolbar, click the Radar Coverage button ().
- On the Display Settings dialog box, Observer Settings page, select the Radar Coverage check box.

9.7.2 Displaying Radar Coverage

To display radar coverage, you must enable it for the observer and then display it for a specific simulation object. For details about enabling radar coverage, see "9.7.1 Enabling the Display of Radar Coverage" above.

➤ **To display radar coverage:**

1. Select a simulation object that has a ground-based radar, such as the Patriot Radar AN/MPQ-65 (entity-level) or Air Defense Radar (aggregate-level).
2. Choose **Objects > Decorations > Pin Radar Coverage**. A radar coverage display is added, centered on the selected simulation object.

Removing Radar Coverage Pins

➤ **To remove the radar coverage pin:**

1. Select a simulation object that has radar coverage pinned.
2. Choose **Objects > Decorations > Pin Radar Coverage**. The radar coverage display is removed.

➤ To remove radar coverage pins for all entities, choose **Objects > Decorations > Remove All Pin Radar Coverage**.

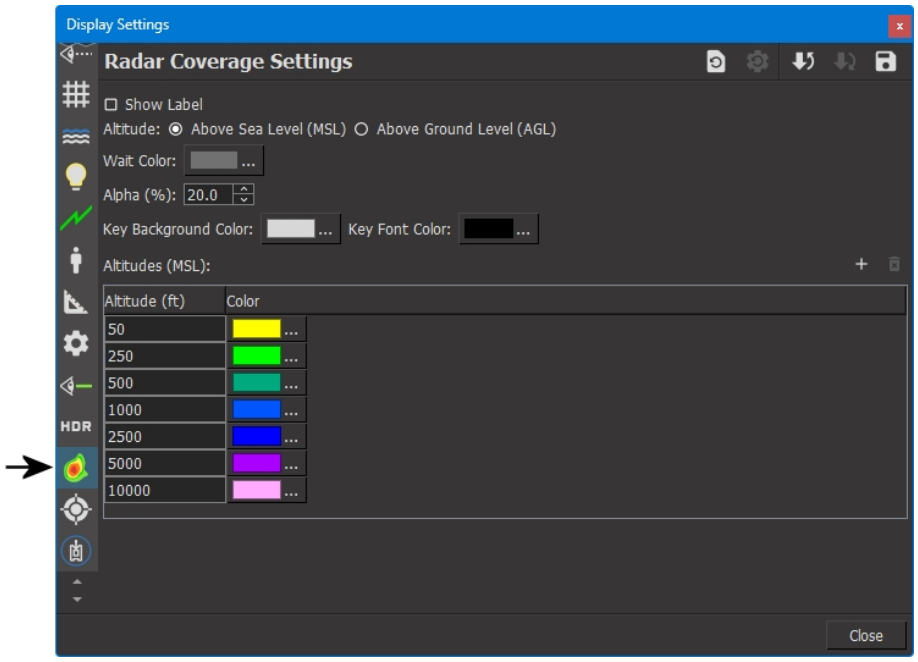
9.7.3 Configuring Radar Coverage Displays

You can configure the colors used for radar coverage displays, the altitude at which colors change, and other aspects of the display.

➤ **To configure radar coverage:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Radar Coverage Settings page (Figure 106).

Figure 106. Radar Coverage Settings page



3. Change the settings as desired. Table 17 describes the settings.

Table 17. Radar coverage settings

Option	Description
Show Label	Display the name of the radar coverage area.
Altitude	Calculate altitude above sea level or above ground level.
Wait Color	The color to use for the radar coverage fan while the GUI waits for results from the sim engine.
Alpha	The amount of transparency for the coverage areas.
Key Background Color	The background color to use for the altitude key.
Key Font Color	The color to use for the text on the altitude key.
Altitudes	The altitudes to test for intersections and their associated colors.

9.8. Using Sonar

VR-Forces simulates active and passive sonar. Entities can be configured with three types of sonar sensor systems: Active Sonar, Passive Sonar, and Sonar. The Sonar system includes both active and passive sonar. Both types of sonar use the VR-Forces sensor signature mechanism. The signatures are different, reflecting the different chances of detecting entities using the two types of sonar.

Passive sonar has these characteristics:

- Sonar can be dipped at a depth other than that of the entity.
- If an entity is moving faster than a specified speed, passive sonar does not work.
- Sonar takes into account thermocline data. Thermoclines are layers of water that have different temperatures and, therefore, different density. Sound transmission varies as it passes through the thermocline. In VR-Forces sonar effectiveness is reduced as it passes through thermocline boundaries. For more information, see "11.9. Setting the Thermocline" on page 347.
- Passive sonar ignores line-of-sight. It does not take into account underwater terrain or the presence of islands.
- If an entity is publishing propulsion noise, it can affect its detectability by passive sonar.

Active sonar has these characteristics:

- Active sonar should acquire targets more quickly than passive sonar.
- If an entity has an active sonar system enabled, it is more easily detected by passive sonar systems.
- Entities with an active sonar that is enabled publish Underwater Acoustics PDUs.
- Entities can have multiple active sonar modes, similar to radar modes, using emitter beams. For details about configuring emitters, see "80.1. Configuring Emitters" on page 1572.
- Active sonar detects targets within the range of its emitter beams, but without regard to the direction of the beams.
- Active sonar ignores propulsion noise and the presence of active sonar on the target.

Sonar only works in water. Therefore rotary-wing entities with sonar systems must set their depth in order to sense anything.

9.8.1 Propulsion Noise and Sonar

If an entity is generating propulsion noise, it is easier to detect using passive sonar. VR-Forces publishes propulsion noise for surface and subsurface entities. If the power plant for an entity is turned off, it does not publish propulsion noise. To turn propulsion noise on or off, change the power plant appearance (for details, see "40.8. Appearance" on page 869) or set the **power-plant-active** parameter in the propulsion-noise-generator component of the entity's movement system.

VR-Forces publishes three propulsion noise values:

- Current Shaft RPM, calculated as **current speed * speed-to-rpm-factor**.
- Ordered Shaft RPM, calculated as **ordered speed * speed-to-rpm-factor**.
- Shaft RPM Rate of Change, calculated as **current acceleration * acceleration-to-rpm-factor**.

The values for current speed, ordered speed, and current acceleration are the current values for the entity. The values for speed-to-rpm-factor and acceleration-to-rpm-factor are taken from the **speed-to-rpm-factor** and **acceleration-to-rpm-factor** parameters in the propulsion-noise-generator component of the entity's movement system.

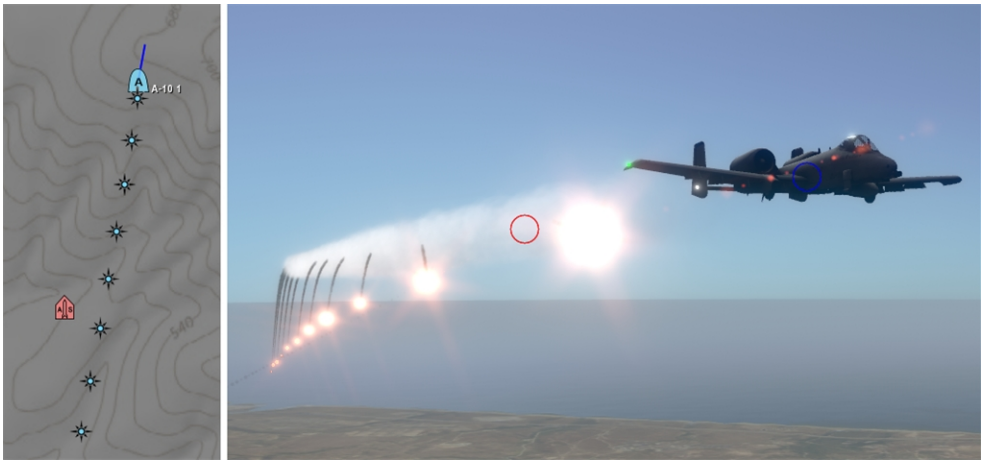
When a target is being detected by passive sonar, if it is publishing propulsion noise, its calculated RPM is compared to a nominal RPM (default 100) to generate a sensor signature modifier. For example, if an entity is publishing an RPM of 80, its signature is scaled by 0.8. If its RPM is 150, the signature is scaled by 1.5.

9.9. Launching Countermeasures (Chaff and Flare)

Fixed-wing and rotary-wing entities can launch countermeasures (chaff and flare) when they are targeted by air-to-air and ground-to-air missiles. By default, entities launch countermeasures automatically based on settings in the *countermeasures.sysdef* file. When an aircraft detects a hostile missile within a specified range, it determines the type of missile and looks up the type of countermeasures to use in the *USCounterMeasuresTable.asl* or *CISCounterMeasuresTable.asl* table.

Figure 107 shows a friendly force fighter launching countermeasures against a sidewinder missile.

Figure 107. Countermeasures (flare)



You can disable automatic countermeasures with the Countermeasures Auto Launch set data request. You can also launch countermeasures with the Launch Countermeasures task. The Launch Countermeasures task is not an exclusive task, so it does not interfere with the entity's current task. For details about the task and set data request, see "31.25. Launch Countermeasures" on page 640 and "40.20. Countermeasures Auto Launch" on page 874.

You can configure the behavior of automatic countermeasures in the *counter-measures-launcher.sysdef* and *cis-counter-measures-launcher.sysdef* files (in *./data/simulationModelSets/EntityLevel/vrfSim/systems/other*). Table 18 describes the parameters that you might want to change.

Table 18. Countermeasure launcher parameters

Parameter	Description
auto-launch-enabled True	Enables or disables automatic launching of countermeasures.
auto-launch-range	The range within which an entity will detect missiles against which to launch countermeasures.
auto-launch-reset-time	The amount of time, in seconds, an entity will wait before launching additional countermeasures.
auto-launch-default-resource	The default countermeasure to use.
auto-launch-number-of-resources	The number of countermeasures to launch for each launch sequence.
counter-measures-select-table-file	The countermeasures selection table to use to determine which countermeasures to fire for a particular missile.

9.10. Modeling Artillery Munitions

In entity-level scenarios, VR-Forces models artillery in two ways: it models an artillery entity that you have tasked to fire, or it models an indirect fire event in an area you specify. When you create a task directing a weapon such as a howitzer to fire munitions, VR-Forces models both the artillery entity and the munitions it fires. Indirect fire events simply model artillery barrages within an area, not the artillery entities or munitions. For details about indirect fire events, see "Chapter 10. Indirect Fire, Ballistic Missiles, and Munition Detonations" on page 313.

By default, when an artillery entity fires, VR-Forces models the munition. Munitions are not affected by wind speed or friction. VR-Forces can also model artillery without modeling the munition. In this case, it just publishes a fire event and detonation event. Modeling munitions requires more computer resources than not doing so. Therefore, in large simulations you may want to disable modeling of munitions.

To disable modeling munitions, in the indirect-fire-system, set the **simulate-munition** parameter to false for each weapon system by editing the sysdef file.



10. Indirect Fire, Ballistic Missiles, and Munition Detonations

This chapter explains how to create and edit indirect fire events and ballistic missiles. These features are only supported in entity-level scenarios.

10.1. Introduction to Indirect Fire 314
- 10.1.1 Creating an Indirect Fire Event 314
- 10.1.2 Editing Indirect Fire Events 318
- 10.1.3 Deleting Indirect Fire Events 318
- 10.1.4 Configuring Indirect Fire Event Default Values 318

10.2. Ballistic Missiles 319
- 10.2.1 Firing Ballistic Missiles 320
- 10.2.2 Editing Missile Target Events 321
- 10.2.3 Deleting Missile Target Events 322

10.3. Creating Munition Detonations 322

10.1. Introduction to Indirect Fire

Indirect fire events allow you to create, configure, and schedule indirect fire. Indirect fire events are not associated with a particular simulation object such as a howitzer. VR-Forces does not model the launcher or trajectory of the munitions. Indirect fire results in one or more salvos of one or more rounds in or near a designated area of the terrain. Figure 108 illustrates indirect fire within an indirect fire area.

Figure 108. Indirect fire event



Indirect fire types are configured in *./data/simulationModelSets/EntityLevel/indirectArtilleryTypes.mtl*.

Indirect fire events, which are area-based, are not supported in aggregate-level scenarios. However, there is an indirect fire task available to artillery entities in aggregate scenarios; see "35.2.13 Indirect Fire" on page 718 for details.

10.1.1 Creating an Indirect Fire Event

When you create an indirect fire event, you specify an area in which the event is to take place, the parameters of the event, and a starting time. The area in which an indirect fire event takes place is a tactical graphic and is listed on the Objects List View of the Objects List Panel.

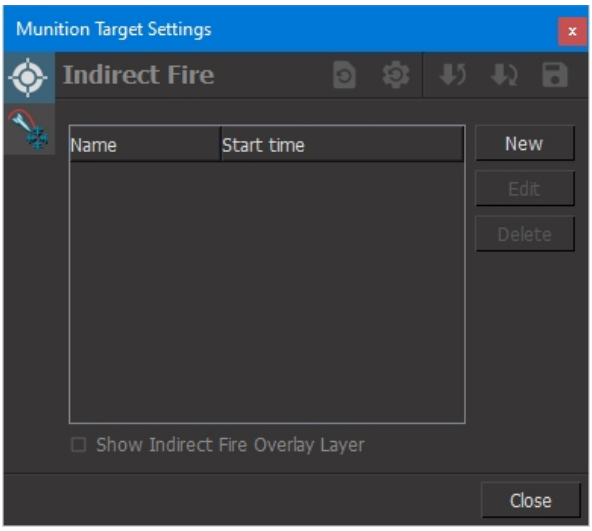
By default, indirect fire areas are placed on the Indirect Fire layer (which is created for the first indirect fire object). (For information about overlays, see "Chapter 46. Overlays" on page 1003.)

You can activate and deactivate indirect fire areas; see "47.11. Active and Inactive Tactical Graphics" on page 1035 for details.

➤ **To create an indirect fire event:**

1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens (Figure 109).

Figure 109. Munition Target Settings dialog box, Indirect Fire page



2. Select the Indirect Fire page.
3. Click New. The cursor changes to create mode and a tab for the Create Indirect Fire dialog box is added to the window.
4. Draw an ellipse on the map to specify the area of the fire event:
    a. Click to set the center of the ellipse. A center point is set and a small circle is displayed with a vertex indicator at the bottom of the circle.
    b. Drag the mouse up or down and then click to set the height of the ellipse. A vertex indicator is added to the left side of the circle.
    c. Drag the mouse left or right and then click to set the width of the ellipse (Figure 110).

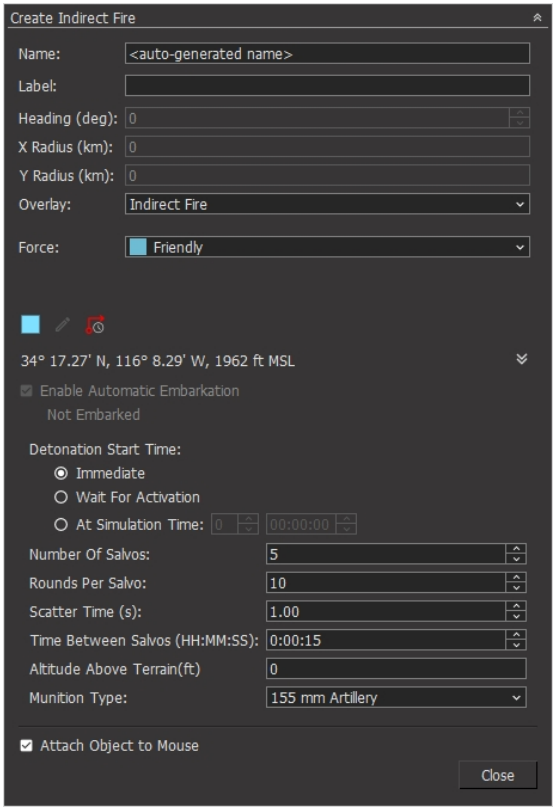
    Optionally, you can specify the properties of the ellipse in the Create Indirect Fire dialog box.

Figure 110. Indirect fire event area



5. Click the tab to display the Create Indirect Fire dialog box (Figure 111).

Figure 111. Create Indirect Fire dialog box

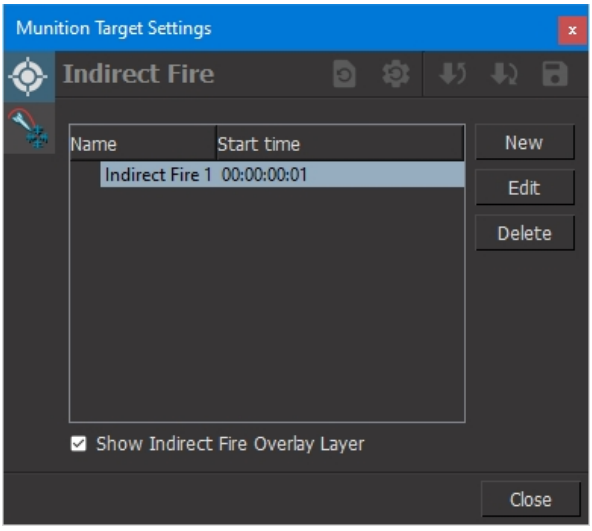


6. Enter a name for the event in the Name text box, or accept the default name (Indirect Fire *n*).
7. Optionally, specify an overlay layer for the event. (If there are no layers in the scenario, one is created when you click Create to create this indirect fire event.)
8. Configure the parameters of the fire event:

- Select the detonation start time. The options are:
  - Immediate. Indirect fire begins immediately.
  - Wait for Activation. The indirect fire area is created in the inactive state. Indirect fire begins when an Activation set data request activates the area. This allows you to activate indirect fire manually or from a plan.
  - At Simulation Time. The simulation time at which you want the munition to start firing.
- In the Number of Salvos box, enter the number of salvos you want to fire.
- In the Rounds Per Salvo box, enter the number of rounds you want to fire for each salvo.
- In the Scatter Time box, enter the length of time for each salvo. The rounds in each salvo will be detonated at random times within this time period.
- If you specified more than one salvo, in the Time Between Salvos box, enter the amount of time you want to pass between each salvo.
- Optionally, specify an altitude above the terrain at which the rounds should detonate.
- Select the munition type in the Munition Type list.

9. Click Create. The Indirect Fire event is added to the list (Figure 112). The area is added as an object on the Indirect Fire layer on the Overlays View of the Objects List Panel (unless you chose a different layer).

Figure 112. Indirect Fire page with indirect fire event



10.1.2 Editing Indirect Fire Events

You can edit the properties of an indirect fire event. You can also directly manipulate the area that specifies the event.

➤ **To edit an indirect fire event:**

1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens.
2. Select the Indirect Fire page (Figure 112 on the previous page).
3. Select the indirect fire event that you want to edit.
4. Click Edit. The Edit Indirect Fire dialog box opens.
5. Change the properties that you want to change.
6. Click OK.

10.1.3 Deleting Indirect Fire Events

➤ **To delete an indirect fire event:**

1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens (Figure 109 on page 315).
2. Select the Indirect Fire page (Figure 112 on the previous page).
3. Select the indirect fire event that you want to delete.
4. Click Delete.
5. Click Close.

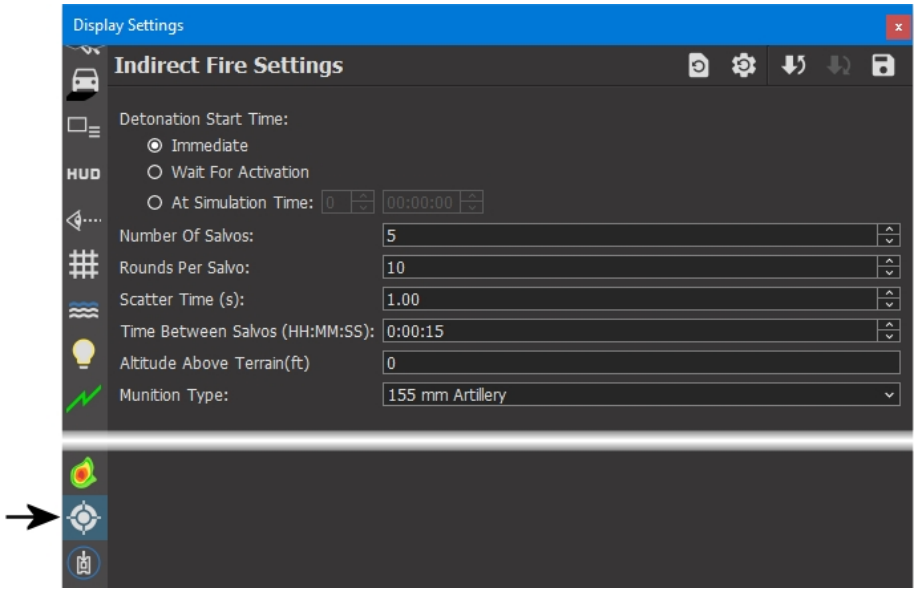
10.1.4 Configuring Indirect Fire Event Default Values

You can configure a set of default values that are applied to the Create Indirect Fire dialog box.

➤ **To configure default values for indirect fire events:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Indirect Fire Settings page (Figure 113).

Figure 113. Indirect Fire Settings page



3. Set the parameters to the desired default values. Most are self-explanatory. The Add Detonations if Missing check box causes VR-Forces to map a detonation to a detonation graphic if there are no mappings for that detonation event in the Detonation Mapping Settings page of the Simulation Object Type Mappings dialog box. If you do not want a detonation displayed for detonation events that are not mapped, clear this check box.
4. Click Close.

10.2. Ballistic Missiles

VR-Forces supports two types of ballistic missiles:

- Strategic missile weapon systems. Entities with strategic missile weapon systems can fire at targets using Launch Ballistic Missile tasks.
- Fixed trajectory ballistic missiles. The trajectory of a missile is determined by a series of coordinates in a comma-separated values (CSV) file. VR-Forces includes some sample CSV files. VR-Forces users are responsible for creating their own files if they want missiles to fly with different characteristics.

Strategic missile systems calculate a trajectory based on a maximum altitude and the distance to the target. They attempt to accurately represent the speed of the missile and they can take into account the curvature of the earth. (For information about strategic missile launch tasks, see "31.23. Launch Ballistic Missile at Location" on page 639 and "31.24. Launch Ballistic Missile at Target Object" on page 639.)

The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

You fire a fixed trajectory ballistic missile by specifying a target and choosing a missile type. The characteristics of the missile type are specified in the Simulation Object Editor. A missile launches at the same altitude as the target point at a distance determined by the CSV file. Therefore, if a target point is placed at or near the ground, it is possible that due to variations in the terrain, the launch location could be underground. Once the missile launches, it follows the trajectory contained in the CSV file that it is configured with. It detonates when it reaches the target point. For details about configuring missiles, see "70.8.1 Configuring Ballistic Missile Movement" on page 1399.

This remainder of this section describes how to fire fixed trajectory ballistic missiles.

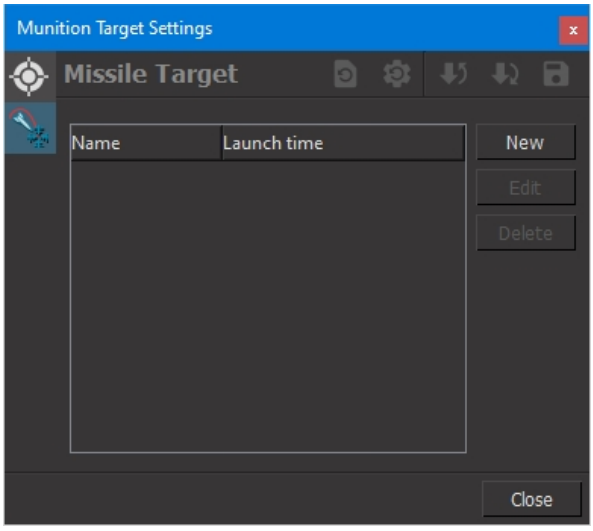
Ballistic missiles are not supported in aggregate-level scenarios.

10.2.1 Firing Ballistic Missiles

➤ **To fire a ballistic missile:**

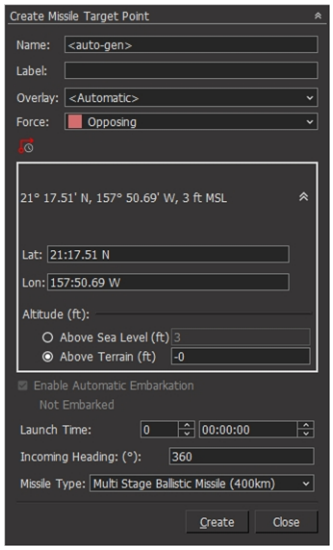
1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens.
2. Select the Missile Target page (Figure 114).

Figure 114. Munition Target Settings dialog box, Missile Target page



3. Click New. The cursor changes to cross-hairs.
4. Click the location where you want the missile to hit. The Create Missile Target dialog box opens (Figure 115).

Figure 115. Create Missile Target dialog box



5. Optionally, specify the parameters for the target. In particular, set these parameters:
   - Name. The name of the target point.
   - Launch Time. The simulation time at which the missile will be launched.
   - Incoming Heading. The heading, relative to the target, from which the missile will be launched.
   - Missile Type. An option from the list.
6. Click Create. The missile event is listed in the Missile Target window.

10.2.2 Editing Missile Target Events

You can edit the properties of a missile target event. You can also directly manipulate the target point.

➤ **To edit a missile target event:**

1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens (Figure 109 on page 315).
2. Select the Missile Target page (Figure 115 above).
3. Select the missile fire event that you want to edit.
4. Click Edit. The Edit Missile Target dialog box opens.
5. Change the properties that you want to change.
6. Click Update.

10.2.3 Deleting Missile Target Events

➤ **To delete a missile target event:**

1. Choose **Simulation > Munition Targets**. The Munition Target Settings dialog box opens (Figure 109 on page 315).
2. Select the Missile Target page (Figure 115 on the previous page).
3. Select the missile fire event that you want to delete.
4. Click Delete.
5. Click Close.

ⓘ You can also delete a target event by deleting the target point from the terrain.

10.3. Creating Munition Detonations

When weapons fire munitions, they detonate and cause damage to simulation objects, terrain, or both. You can also manually create munition detonations. They have the same effect as the simulated munitions they represent.

ⓘ Munition detonations are not supported in aggregate-level scenarios.

➤ **To create a munition detonation:**

1. Choose **Create > Munition Detonation**. The Create Detonation dialog box opens, as if you were creating a simulation object or tactical graphic. The cursor changes to red cross-hairs.
2. If you want to use the currently configured detonation type and result, click on the terrain where you want the detonation to occur. You can continue to click to create more detonations while in this mode.
3. If you want to change the munition type or result:
    a. In the Munition Type list, select the type of munition you want to detonate.
    b. In the Result list, select the type of detonation that you want to occur.
    c. Optionally, specify the location for the detonation.
    d. Click on the terrain where you want the detonation to occur. Click as many times as you want.
4. To exit detonation mode, close the Create Detonation dialog box or press **Esc**.



11. Setting Environment Conditions

This chapter explains how to set the time of day and weather conditions.

> If you want environment conditions in different applications or instances to look the same, use the command-line option (`--hasNTPSync | -O`). This enables the synchronization of all time-based features between different VR-Vantage applications, such as instances of VR-Vantage or the VR-Forces GUI. Note that system clocks must be synchronized.

11.1. Introduction to Environment Conditions 324
11.2. Setting Weather Conditions 324
11.2.1 Setting the Sky Cover, Wind Direction and Speed 324
11.2.2 Setting the Air Pressure and Temperature 327
11.2.3 Setting Visibility 327
11.2.4 Specifying Precipitation Type and Intensity 328
11.2.5 Setting Rain Accumulation 329
11.2.6 Setting Snow Depth and Elevation 329
11.3. Setting the Date and Time of Day 330
11.3.1 Setting the Date 331
11.3.2 Setting the Time of Day 332
11.3.3 Choosing the Illumination Model 332
11.4. Selecting an Environment Condition 333
11.4.1 Adding an Environment Condition 335
11.4.2 Editing an Environment Condition 335
11.4.3 Deleting an Environment Condition 335
11.5. Creating Local Weather Zones 336
11.5.1 Creating a Weather Zone Simulation Graphic 337
11.5.2 Editing a Weather Zone's Environment 338
11.6. Using Weather Simulation Objects 339
11.6.1 Creating a Wind Corridor 339
11.6.2 Creating an Air Turbulence Area 340

11.7. Displaying Weather Conditions at a Location 340
11.8. Marine Conditions 341
11.8.1 Enabling Marine Effects 342
11.8.2 Configuring Advanced Marine Conditions 343
11.8.3 Enabling Tidal Stream Wakes 345
11.8.4 Setting Ocean Quality 346
11.8.5 Specifying an Ocean Wave Spectrum 346
11.8.6 Configuring Breaking Waves 347
11.9. Setting the Thermocline 347
11.10. METOC Object Classes that Can Be Published 348

11.1. Introduction to Environment Conditions

You can set the date, time of day, weather, and marine conditions from the GUI and in plans. The date, time of day, and weather conditions are sim engine settings, not merely GUI display effects. They affect simulation object sensors and line-of-sight. For example, if visibility is set to 4 km, a simulation object will not see a simulation object that is 5 km away. If the time of day changes from daytime to nighttime, lighting conditions in the 3D view change accordingly.

Date and Time and Weather are options on the Commands menu in plans.

11.2. Setting Weather Conditions

You can set these weather conditions:

- Cloud cover
- Visibility
- Precipitation
- Wind speed and direction
- Ambient air temperature

When you specify weather, the sim engine sends out state updates with these conditions. Simulation objects respond to them only if their model has the ability to do so.

11.2.1 Setting the Sky Cover, Wind Direction and Speed

You can set atmospheric conditions, including wind speed and direction, visibility, obscurant (such as fog, smoke, or dust), cloud types, and sky coverage. If you do not specify the sea state manually, it is set based on the wind conditions. (The maximum wind speed that you can set is 360 km/h.) The wind direction affects the angle at which precipitation falls. Wind speed and direction also affect SpeedTrees, grass, and particle systems (dust clouds, smoke, and so on).



Local weather zones are not affected by global wind speed. They use the wind speed and direction specified in the area's Weather settings.

> The arrow in the dial indicating the wind direction points in the direction that the wind is moving. For example, if the wind direction is set to 180 degrees, the arrow points up to show that the wind is moving from the south to the north.

On the Weather page, you can specify different wind speeds and directions, visibility, obscurants, and cloud characteristics at various altitudes. This allows for a more realistic simulation for aircraft and simulation objects with parachutes. You can create up to ten levels of atmospheric configurations.

For information on setting the color of ground-level fog, see "Configuring Ground Fog" on page 328.

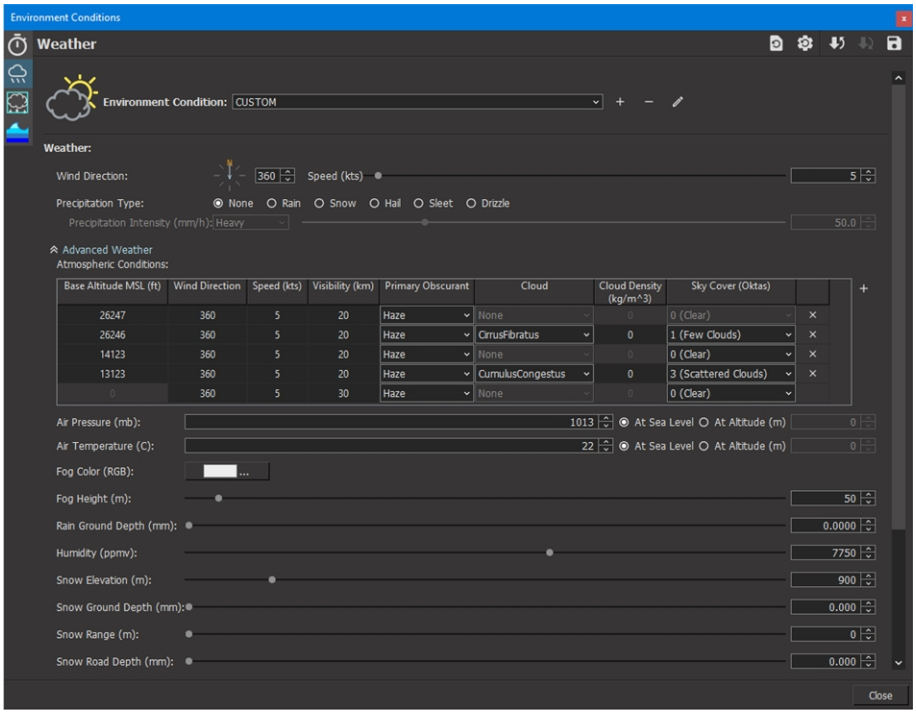
The highest elevation in the Atmospheric Conditions table (under Advanced Weather) is automatically configured with a Sky Cover value of 0 (clear) with no clouds. This elevation's Base Altitude MSL value provides an effective ceiling for the clouds at the elevation immediately underneath it. You cannot change the sky cover for this layer; you must add or edit lower layers.

If you set Visibility to a large value, you might need to adjust the value of the channel's Z far attribute (also known as the far clipping plane). If the Z far attribute is set to a value smaller than the Visibility value, objects beyond the Z far distance are not visible in the channel. For details, see "83.4.1 Setting the Clipping Planes" on page 1605.

➤ **To set wind, visibility, and other elevation-based atmospheric conditions:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116).

Figure 116. Weather page



3. Click Advanced Weather to expose the Atmospheric Conditions table.
4. Set the atmospheric conditions for the desired levels of elevation:
    a. Select an environment condition from the Environment Conditions list, or click the Add button (+) next to the list to create a new environment condition.

       The lowest elevational configuration in the Atmospheric Conditions table begins at 0 feet above mean sea level (MSL). You can specify the base of each subsequent elevation.

    b. Use the Wind Direction dial in the Weather section of the page to enter a Wind Direction heading between 001 and 360 degrees.

    c. Drag the Speed slider to set the wind's speed.

       The wind heading and speed you entered appear in the Atmospheric Conditions table.
5. Set the visibility, obscurant, and sky cover for one or more levels in the Atmospheric Conditions table:
    a. Select Visibility in a row in the table or click the Add button (+) to the right of the table to create a new row.

    b. Enter a Visibility distance.

    c. If visibility is limited, choose the Primary Obscurant.

    d. Set the Sky Cover.

e. If you set Sky Cover to a nonzero value, you activate the Cloud column. Choose a cloud type.

f. If you set Visibility to a high value, you might need to change the channel's Z far attribute value — it must be higher than the Visibility value. Z far defaults to 200 kilometers. For details on changing Z far, see "83.4.1 Setting the Clipping Planes" on page 1605.

11.2.2 Setting the Air Pressure and Temperature

You can specify the air pressure and temperature at any altitude, including sea level. When the air temperature is below freezing, the ground simulates freezing and affects ground vehicle mobility. VR-Forces does not do anything with the air pressure information. It is sent over the network for the use of applications that need to model air pressure.

➤ **To set the ambient air temperature:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on the previous page).
3. In the Weather section of the page, click Advanced Weather. The dialog box expands to show more options.
4. In the Air Pressure field, set the air pressure in millibars.
5. Select At Sea Level or At Altitude.
6. If you selected At Altitude, set the altitude in meters for the new air pressure. (Setting an altitude of 0 has the same effect as selecting At Sea Level.)
7. In the Air Temperature field, set air temperature in degrees Celsius.
8. Select At Sea Level or At Altitude.
9. If you selected At Altitude, set the altitude in meters for the new air temperature. (Setting an altitude of 0 has the same effect as selecting At Sea Level.)

11.2.3 Setting Visibility

The visibility setting determines the denseness of obscurants in the atmosphere (fog, smoke, or dust, for example), which determines how far you can see.

In the GUI, there is no difference between the appearances of the various obscurants. VR-Forces publishes the information to the network.

➤ **To set visibility:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on the previous page).
3. In the Atmospheric Conditions table, change the value in the Visibility column.
4. If visibility is limited, choose a Primary Obscurant.

5. If you set Visibility to a large value, you might need to adjust the value of the channel's Z far attribute (also known as the far clipping plane). If the Z far attribute is set to a value smaller than the Visibility value, objects beyond the Z far distance are not visible in the channel. For details, see "83.4.1 Setting the Clipping Planes" on page 1605.
6. To change visibility at a different elevation, click the Add button (+) and then specify the values for that elevation.

Configuring Ground Fog

You can set the height and color of ground-level fog. The fog height affects how much sky the observer can see when looking up from ground level. For example, if you set the fog height to 50m and look up, you will see some amount of sky above you, and to the sides it will be very gray. If you then set it to 10m and look up, you will see mostly blue sky above you, but still quite a bit of gray to the sides.

These steps create ground-based fog and let you choose the fog's color. To configure fog at higher elevations (or other obscurants like smoke or dust), see "11.2.3 Setting Visibility" on the previous page. To configure fog (or other obscurants) that's confined to a limited area, see "11.6. Using Weather Simulation Objects" on page 339.

> Fog settings are not modeled in the sim engine. Therefore, they are not available when you set weather conditions in a plan.

➤ **To configure fog:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on page 326).
3. In the Weather section of the page, click Advanced Weather. The dialog box expands to show more options.
4. To set the fog height, adjust the slider or enter a value in the box.
5. To set the fog color:
    a. Click the color swatch. A color picker dialog box opens.
    b. Select the color you want to apply to the fog. It is applied immediately, so you can easily see the effect of the new color and change it if you want to.
    c. Click OK.

11.2.4 Specifying Precipitation Type and Intensity

VR-Forces supports display of rain, snow, hail, sleet, and drizzle. You can configure the intensity of precipitation.

> Precipitation has no effect on the simulation. It is only a visual effect. Snow does not accumulate.

➤ **To specify the precipitation type and intensity:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on page 326).
3. Select a Precipitation Type. Once you select a type other than none, the Precipitation Intensity setting becomes editable.
4. Adjust the Precipitation Intensity slider or enter a value in the box. The range is 0 through 200 mm per hour. The intensities list updates based on the rate you specify. You can also choose one of the preset intensities from the list:
   - Light – 1.2 mm per hour.
   - Moderate – 6.2 mm per hour.
   - Heavy – 30 mm per hour.
   - Extreme – 125 mm per hour.

11.2.5 Setting Rain Accumulation

You can specify the accumulation of rain on the terrain. When you have rain accumulation, the terrain appears slightly shinier and specular.

You can also specify that puddles should appear in areas of the terrain. If it is raining, ripples appear in the puddles. To define where puddles should accumulate, use a Rain Texture map. See "Using Shader-based Effect Maps" in *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To set the rain accumulation:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on page 326).
3. Click Advanced Weather.
4. Adjust the Rain Ground Depth slider or enter a value in the box. This specifies the depth of rain on the ground and other terrain features. The depth specifies the level of specular reflection on the ground and the appearance of puddles.

11.2.6 Setting Snow Depth and Elevation

You can specify the depth of snow on the ground and roads, and you can also specify the elevation for the snow line. Snow can also display on the roofs of buildings.

If appropriate, you can mask out the snow for certain parts of the terrain using a Snow Texture map. You can also use the SNO effect texture or the snow mask effect map layer to have snow drift across the roads when the wind is blowing. See "Using Shader-based Effect Maps" in *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To set the snow depth and minimum elevation:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 116 on page 326).
3. Click Advanced Weather.
4. Adjust the Snow Ground Depth slider or enter a value in the box. This specifies the depth of the snow on the ground and other terrain features. The range is 0.000 through 10,000.000 mm.
5. Adjust the Snow Road Depth slider or enter a value in the box. This specifies the depth of the snow on roads. The range is 0.000 through 10,000.000 mm.
6. Adjust the Snow Elevation slider or enter a value in the box. This specifies the minimum elevation (in meters) where the snow cover appears.
7. If desired, use the Snow Range slider and box to specify the range in elevation at which the snow transitions from no snow to the defined snow depth. Adjust the slider or enter a value.

   For example, if the Snow Elevation is set to 100 m and the range is set to 50 m, there will be no snow at 100 m, half the current snow coverage at 125 m, and full coverage at 150 m.
8. If you want to see snow blowing in the wind, create a Weather tactical graphic and specify the Blowing Snow environmental condition. See "11.5. Creating Local Weather Zones" on page 336.
9. Click Close.

11.3. Setting the Date and Time of Day

You can set the date, time zone, and time of day.

- Date: When you create a scenario, you specify a date. However, you can change the date at any time while you are creating or running a scenario. The date at a location is advanced when the time at that location advances through midnight. When this happens, all locations to the east, up to the time zones at UTC + 12 hours, are in the advanced date; all locations to the west, up to the time zones at UTC -12 hours, are in the previous date. Thus there is effectively a date line at a longitude 12 hours from UTC.
- Format: You can specify whether the time and date should be display in standard format or military format. You can also choose a display format for the date and time, which will be one of a variety of options relevant to either standard or military.
- Time zone: You can change the time zone displayed in the Environmental Settings toolbar to any zone that you want. If you change the time zone in the Environment Conditions dialog box, the time displayed in the Environment Settings Toolbar changes to the current time for that time zone. However, the time at a given location on the earth does not change. The illumination at a location therefore does

not change when the time zone displayed in the toolbar changes. Since the time zone at a location may not be the same as the time zone displayed in the toolbar, the illumination at that location may not match the time displayed in the toolbar.

- Time of day: The time of day advances as simulation time advances. However, they are not directly tied to each other. For example, if at the start of a simulation (simulation time 0:00:00:00), you set the time of day to 10:00 AM, after eight hours of simulation time passes, the time of day will be 6:00 PM. However, you could just as easily set the initial time of day to some other time or change it at will during the course of the simulation. These changes would not affect the elapsed simulation time in any way. They would only change the starting point for the advance of the time of day.

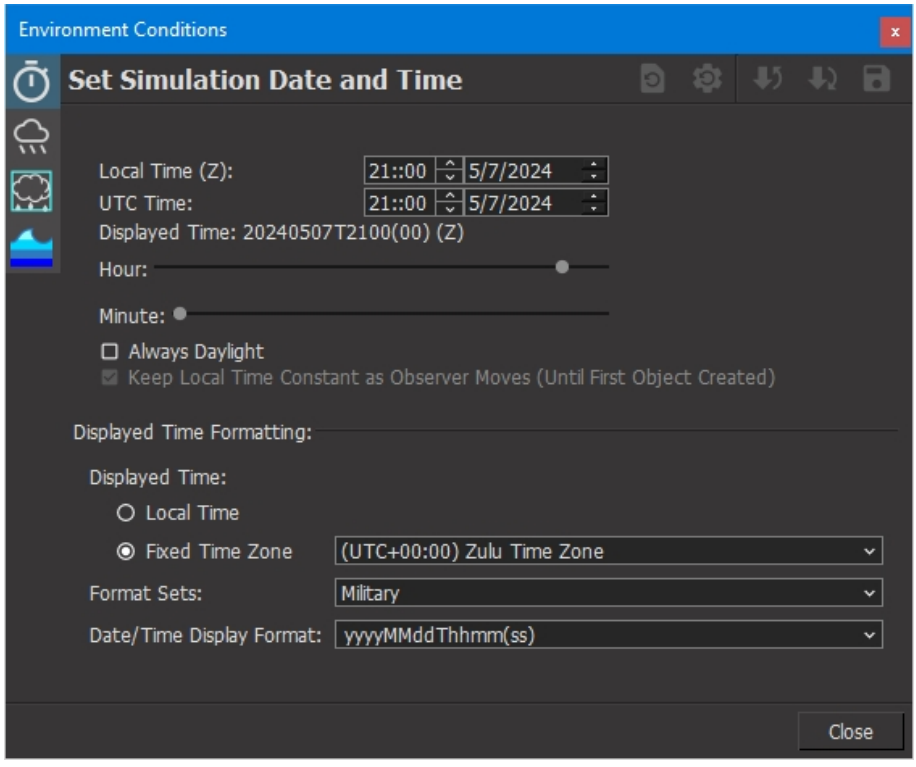
  VR-Forces uses an ephemeris model to calculate illumination based on the date and time of day. The amount of light affects visual sensors and scene lighting in Stealth view mode. Illumination at a location is based on the time of day at that location.

11.3.1 Setting the Date

➤ **To set the date:**

1. Choose **Simulation > Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 117).

Figure 117. Date and Time page



3. Place the cursor in either the Local Time or UTC Time box for the date.

4. To change the month, click the right or left arrows, or click the month and select a month from the list.
5. To change the year, click the right or left arrows, or click the year and type in a year or use the spin controls to select a value.
6. To specify the date, click on the date in the calendar.
7. To specify the format, select the desired options from the Formats Set and Date/Time Display Formats lists.

11.3.2 Setting the Time of Day

As time advances, the time displayed on the Environment Settings Toolbar changes. The time of day setting is independent of simulation time. The default time zone is Coordinated Universal Time (UTC, also called Zulu time). You can change the time zone to the local zone or any of the zones in the Fixed Time Zone list.

If you want to set the time of day and time zone, set the time zone first.

➤ **To set the time of day:**

1. Choose **Simulation > Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 117 on the previous page).
3. Set the time of day by changing the value in the Current Time box or by adjusting the Hour and Minute sliders.
4. Select the time zone in one of the following ways:
   - Select a time zone in the Fixed Time Zone list.
   - Select Local Time. The time zone set for your computer is used.
5. In the Formats Set box, specify the format for the time and date by selecting Military or Standard.
6. Optionally, select a date/time display format. This changes the format used in the Environment Settings Toolbar. The available formats are related to the option the in Formats Set box.

11.3.3 Choosing the Illumination Model

Illumination affects visual sensors. Illumination can be on all of the time or it can be affected by the time of day. The illumination setting does not affect the GUI. It only affects sim engine calculations. If the day/night illumination model is active (Always Daylight is not selected), the sim engine calculates illumination based on a simulation object's location given current Greenwich Mean Time, offset by the time zone where the simulation object is located (based on longitude). This calculation does not take into account daylight saving time or localized time zone variations.

➤ **To specify the illumination model:**

1. Choose **Simulation > Environment Conditions**, or click the date on the Environment Settings Toolbar. The Environment Conditions dialog box opens.
2. Select the Date and Time page (Figure 117 on page 331).
3. Select or clear the Always Daylight check box.

11.4. Selecting an Environment Condition

VR-Forces has a set of preconfigured environment conditions. Each environment condition has settings for:

- Weather:
  - Wind speed and direction.
  - Precipitation type and intensity.
  - Visibility.
  - Primary obscurant (for example, haze, dust or smoke).
  - Cloud type and sky cover.
  - Air pressure.
  - Ambient air temperature.
  - Fog height and color.
  - Rain depth on the ground.
  - Humidity
  - Snow depth on the ground and roads and starting elevation for accumulation.
- Marine:
  - Sea state.
  - Sea swell.
  - Underwater visibility.
  - Choppiness.
  - Depth offset.
  - Surface transparency.
  - Surge depth.
  - Tidal current direction and speed.
  - Wave fetch.

Conditions range from clear to stormy and allow you to quickly set the weather without needing to adjust all the various weather parameters. If you want to, however, you can change any of the individual weather parameters. Changing a weather or marine setting does not affect the saved settings unless you specifically overwrite them.

You can save and import environment settings. Environment settings support the standard VR-Forces settings behaviors, as described in"3.7. Managing VR-Forces Settings" on page 123.

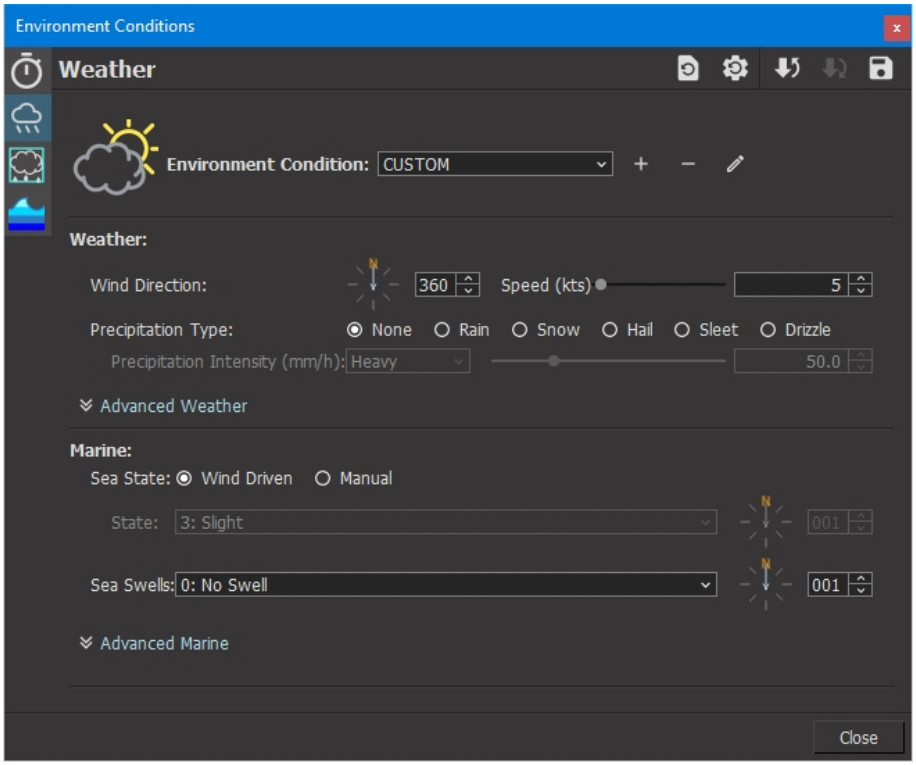
You can define your own environment conditions to include in the list of those you can choose from.

> The settings on the Weather page always reflect the current settings for environment conditions. They do not necessarily match any of the preconfigured environment conditions as listed in the Environment Conditions list except immediately after a preconfigured environment condition is applied. In other words, the Apply Environment Conditions list allows you to apply environment conditions, but it is not intended to summarize the status of the current environment conditions.

➤ **To select an environment condition:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 118).

Figure 118. Weather page



3. In the Environment Condition list, select the condition you want.

> After you select a preconfigured environment condition, if you change any of the settings, the Apply Environment Condition list reads CUSTOM instead of the condition that you chose.

11.4.1 Adding an Environment Condition

If you add an environment condition, it is added to the Apply Environment Condition list.

➤ **To add an environment condition:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 118 on the previous page).
3. Change the weather settings to the ones you want for the new condition.
4. Click the Add button (+). The Name an Environment Condition dialog box opens.
5. Type a name for the new condition.
6. Click OK.

11.4.2 Editing an Environment Condition

You can change the settings for an environment condition. Editing a condition essentially means overwriting a condition with the current set of weather and marine conditions.

➤ **To edit an environment condition:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 118 on the previous page).
3. Optionally, in the Apply Environment Condition list, select a condition you want to use as your starting point.
4. Change whatever conditions you want.
5. Click the Edit button (✎). The Select an Environment Condition dialog box opens.
6. In the Select Name list, select the condition you want to overwrite.
7. Click OK.

11.4.3 Deleting an Environment Condition

You can delete any defined environment conditions. If you delete one of the built-in environment conditions, you can restore it from the factory settings.

➤ **To delete an environment condition:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 118 on the previous page).
3. Click the Delete button (−). The Select an Environment Condition dialog box opens.
4. In the Select Name list, select the condition you want to delete.
5. Click OK.

11.5. Creating Local Weather Zones

You can create local weather zones. Each area can have different weather conditions—for example, rainy, clear, or overcast, with different wind and visibility conditions. Simulation objects can respond to those conditions as they move from one area to another.

Weather zones can be free-form areas, boxes, and circles. You can create them from the Create Object Palette just as you would create the comparable tactical graphics. You can also create box weather zones from the Environment Conditions dialog box on the Local Weather Zones page.

Weather zones have a priority. If two areas overlap and one has a higher priority than the other, a simulation object in the overlapping area responds to the weather conditions in the higher priority area. If overlapping areas have the same priority, the one that is discovered first on the network is used.

By default, weather zones are placed on the Weather overlay. Weather zones display a graphic that represents the weather conditions. Figure 119 illustrates a weather zone set to the Ground Fog environmental condition. You can move and resize weather zones just as you can move and resize other tactical graphics.

Weather zones are used by the simulation engine and affect simulation objects in the scenario.

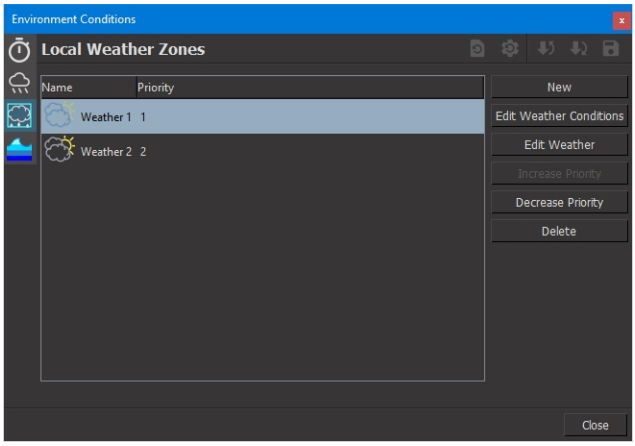
Figure 119. Weather (Area) zone - 2D and 3D



➤ **To create a local weather zone from the Local Weather Zones page:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Local Weather Zones page (Figure 120).

Figure 120. Local Weather Zones page



3. Click New. A weather zone box appears in the scenario.
4. Click on the terrain to place the lower left corner of the weather zone. The Create Weather dialog box opens.

   The default size for a weather zone is 100,000 meters per side, so the zone might entirely cover small terrains. If this is a problem, edit the Length and Width in the Create Weather dialog box to reduce it to a manageable size.

5. Optionally, edit the parameters for the weather zone. In particular, select an environmental condition or edit the environment to specify the conditions you want to apply to this weather zone. For details about editing the environment, see "11.5.2 Editing a Weather Zone's Environment" on the next page.
6. Click Create.
7. On the Local Weather Zones page, select a weather zone and use the Edit Weather Conditions or Edit Weather button to alter the zone's parameters.
8. When weather zones overlap, use the Increase Priority and Decrease Priority buttons to change the priority of the selected weather zone.

11.5.1 Creating a Weather Zone Simulation Graphic

You can create all types of weather zones from the Create Object Palette. Weather zones create effects such as rain, fog, snow, and blowing dust or sandstorms. Precipitation and the Visibility setting in the Atmospheric Conditions affect line of sight because they reduce the apparent signature of entities, thereby reducing the ability of sensors to detect entities. Other effects are not modeled in the sim engine, therefore they do not affect line of sight.

The weather zones are listed as:

- Weather (Area). A free-form shape: you determine the number of sides.
- Weather (Box). The same weather zone you can create from the Local Weather

Zones page.

- Weather (Circle).

The procedures for drawing them are the same as for other similarly shaped graphics, as described in "14.3. Placing Objects" on page 382, "47.1.3 Drawing Circles and Ellipses" on page 1013, and "47.1.7 Drawing Boxes" on page 1016. The main difference is that when you create a weather zone, the Edit dialog box always opens to allow you to specify the weather-specific parameters.

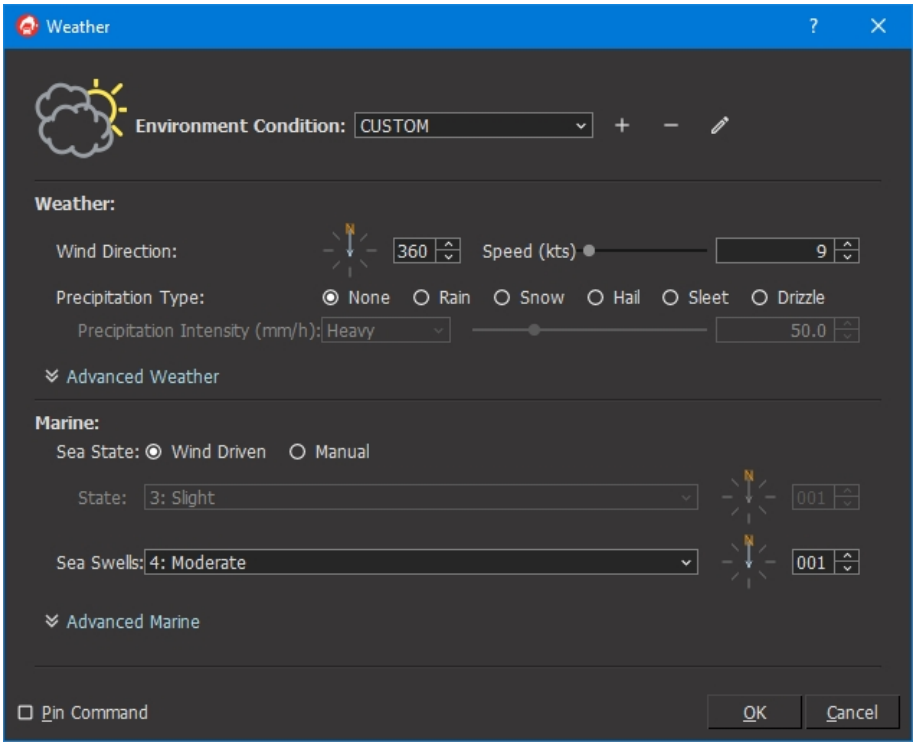
11.5.2 Editing a Weather Zone's Environment

You can edit a weather zone's environment when you create it or after you create it. You can assign it one of the preconfigured environmental conditions or change individual environment settings. You can edit all types of weather zones from the Local Weather Zones page. It does not matter how you created them. For details about preconfigured environmental conditions, see "11.4. Selecting an Environment Condition" on page 333.

➤ **To edit a weather zone's environment:**

1. Do one of the following:
   - In the Create Weather dialog box, click Edit Environment. The Weather dialog box opens (Figure 121).
   - On the Local Weather Zones page of the Environment Conditions dialog box, select the weather zone you want to edit and then click Edit Weather Conditions. The Weather dialog box opens.

Figure 121. Set Weather dialog box



2. Change weather conditions as desired. They are the same as those on the Weather page of the Environment Conditions dialog box. For details, see "11.2. Setting Weather Conditions" on page 324.
3. Click OK.

11.6. Using Weather Simulation Objects

Weather zones, wind corridors, and air turbulence can affect simulation objects. You can also use weather zones for the Entity In Area condition in plans. VR-Forces publishes weather details and VR-Vantage applications can use that information to display them.

Weather tactical graphics include:

- Weather zones (Area, Box, Circle). Areal graphics that visualize localized weather, as described in "11.5. Creating Local Weather Zones" on page 336.
- Wind Corridor. A linear graphic that affects aircraft. See "11.6.1 Creating a Wind Corridor" below.
- Air turbulence. An area that publishes air turbulence to the network. For details, see "11.6.2 Creating an Air Turbulence Area" on the next page.

11.6.1 Creating a Wind Corridor

Wind corridors are linear tactical graphics that specify a directional wind that aircraft can move through or along like a route. You can use this to represent an area where the wind is blowing at a different speed than elsewhere. For example, the wind in a valley might move at a different speed than the wind across the mountains. A wind corridor affects aircraft with either a tailwind or headwind, increasing or decreasing the true airspeed.

As with other linear tactical graphics, wind corridors have a direction. The wind direction starts at the first point you draw and moves to the final point.

➤ **To create a wind corridor:**

1. Open the Create Object Palette.
2. Optionally, click the Weather filter ().
3. Select Wind Corridor.
4. Click to place the vertices of the wind corridor. Double-click when you place the final vertex. The Create Wind Corridor dialog box appears.
5. Specify the parameters for the wind corridor. In addition to the typical parameters, wind corridors have these parameters:
   - Width and height. The dimensions in the X and Z planes.
   - Altitude. This is the altitude of the vertical center of the wind corridor above sea level or terrain. Each vertex in the wind corridor can have a specific altitude. See "14.4. Specifying an Object's Altitude" on page 386 to learn the different ways you can define the altitude.

- Wind speed. The speed of the wind in the corridor.
- Priority. This wind corridor's place in the order of precedence determines which corridor prevails when corridors overlap. The highest priority is 1.

6. Click Create.

11.6.2 Creating an Air Turbulence Area

You can create areas of air turbulence. Air Turbulence Areas allow you designate an area on the network where aircraft may experience turbulence. VR-Forces does not simulate the turbulence, but it is part of the comprehensive weather model simulated by VR-Forces. The information is sent over the network for the use of applications and other simulators that need to model air turbulence.

➤ **To create an air turbulence area:**

1. Open the Create Object Palette.
2. Optionally, click the Weather filter ( ).
3. Select Air Turbulence Area.
4. Click to place the vertices of the area. Double-click when you place the final vertex. The Create Air Turbulence Area dialog box displays.
5. Specify the parameters for the air turbulence area. In addition to the typical parameters, air turbulence areas have these parameters:
   - Floor and Ceiling. The altitude of the bottom and the top of the area.
   - Air Turbulence Level. Select the level of turbulence in the area: Light, Moderate, Severe, or Extreme.
6. Click Create.

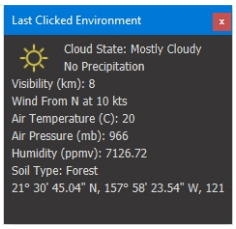
11.7. Displaying Weather Conditions at a Location

VR-Forces can display the weather conditions at a terrain location or the location of an entity in the Last Clicked Environment Panel (Figure 122). If the entity is not on the ground, the panel displays the weather at the altitude of the entity. For subsurface entities, the weather at sea level is reported. The panel shows:

- Cloud cover
- Precipitation
- Visibility
- Wind speed and direction
- Air Temperature
- Air Pressure
- Humidity
- Soil type
- Coordinates and elevation of the location or entity

As the entity moves, the environment information updates every two seconds.

Figure 122. Last Clicked Environment Panel



➤ **To view weather conditions at a location:**

1. Choose **View > Last Clicked Environment Panel**. The Last Clicked Environment Panel opens.
2. Click a location on the map or an entity.

11.8. Marine Conditions

VR-Forces supports a variety of dynamic ocean effects (Figure 123), including:

- Douglas Sea State. This includes wind waves (wind sea), swell character, and the directions of each. VR-Forces allows each to be separately configured.
- Wave chop.
- Surface transparency. The ability to see through the water from above sea level.
- Underwater visibility. The ability to see underwater.
- Swell.
- Surge depth. When bathymetry data is present, this is the depth of the water at which full storm effects take place. Where the water is shallower, it is calmer.
- Tidal current speed and direction. If tidal stream wakes are enabled, they are automatically applied to streamed buoys and beacons.

Figure 123. Ocean effects



- Enabling dynamic ocean effects can affect performance.
- To synchronize the rendering of ocean waves between different products or instances, use a command-line option (--hasNTPSync | -O). Without using that option, waves may not match.

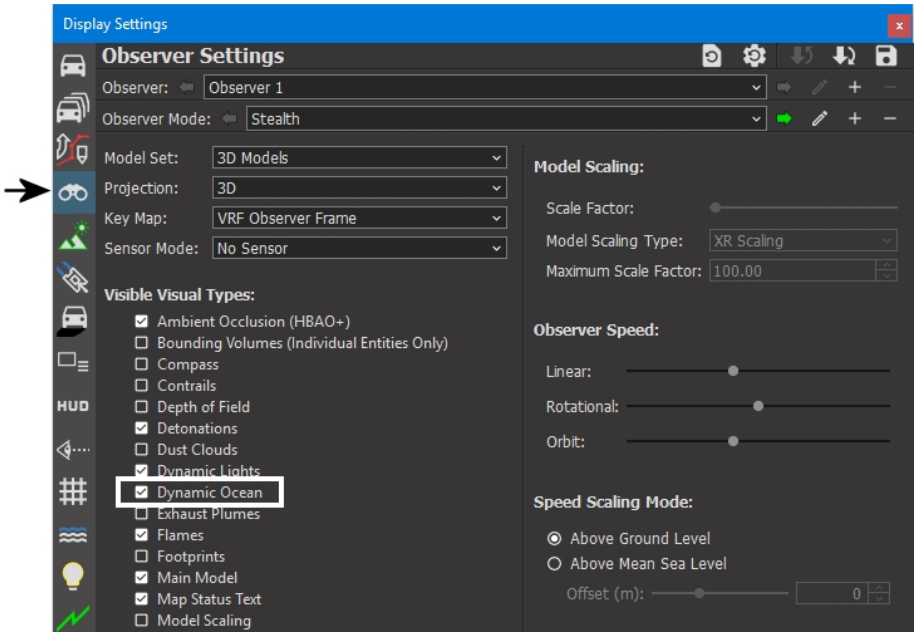
11.8.1 Enabling Marine Effects

Marine effects can reduce performance.

➤ **To enable marine effects (dynamic ocean):**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 124).

Figure 124. Observer Settings page



3. Select or clear the Dynamic Ocean check box.

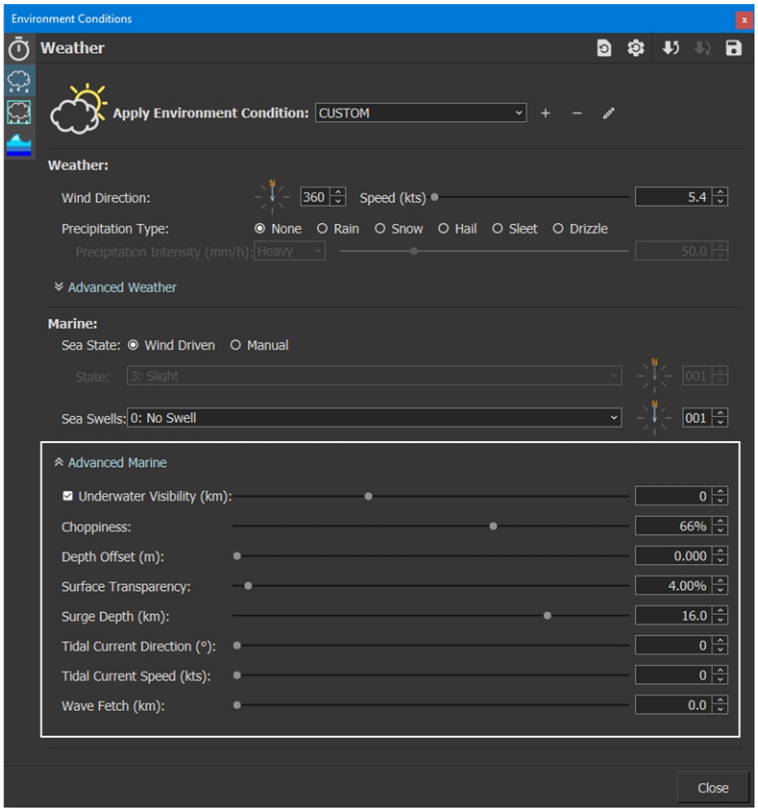
11.8.2 Configuring Advanced Marine Conditions

The preconfigured environment conditions include marine conditions. You can also customize the marine settings at any time.

➤ **To configure marine effects:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Weather page (Figure 125).

Figure 125. Weather page



3. If the Marine section is not expanded, click Advanced Marine. The dialog box expands to show additional configuration options.
4. Change settings as desired. Table 19 describes the marine settings.

Table 19. Advanced marine settings

Parameter	Description
Sea State	Determined by the wind direction and speed or set manually. If manual, select one of the swell states for the Douglas sea state scale. You can also set the compass direction from which sea state originates.
Sea Swells	The amount of rise and fall of waves independent of wind. Choose from 10 fixed swell states. You can also set the compass direction from which swells originate.
Advanced parameters	
Underwater visibility	The distance you can see underwater if the observer is underwater.
Choppiness	The percentage of turbulence on the water.
Depth Offset	The offset of ocean depth.

Table 19. Advanced marine settings (continued)

Parameter	Description
Surface Transparency	The depth at which VR-Forces begins to apply transparency to the water surface if the terrain has bathymetry or underwater polygons. This setting helps reduce Z-fighting along the shoreline. It does not mean that you can see some distance into the water when the observer is over deep water. The maximum value represents transparency to 500 meters.
Surge Depth	When bathymetry data is present, this is the depth of the water at which full storm effects take place. As the water is shallower, it is calmer.
Tidal Current Direction	The direction of the tidal current flow. This affects the display of tidal wakes on buoys and beacons.
Tidal Current Speed	The speed of the tidal current. This affects the display of tidal wakes on buoys and beacons. Default: 0.
Wave Fetch	The wind-fetch distance is the distance that wind can travel over water in a constant direction.

11.8.3 Enabling Tidal Stream Wakes

Tidal stream wakes create wake effects around buoys and beacons based on the movement of the tides. Tidal stream wakes are supported as standalone models. They are automatically attached to streamed buoys and beacons. VR-Forces includes wake models for each of the supported buoy and beacon models. They are named for the buoy or beacon type they support, for example, BuoysPillarWake or BeaconsLatticeWake. You may want to edit the **wakeLength** parameter to provide the best visuals for your terrain.

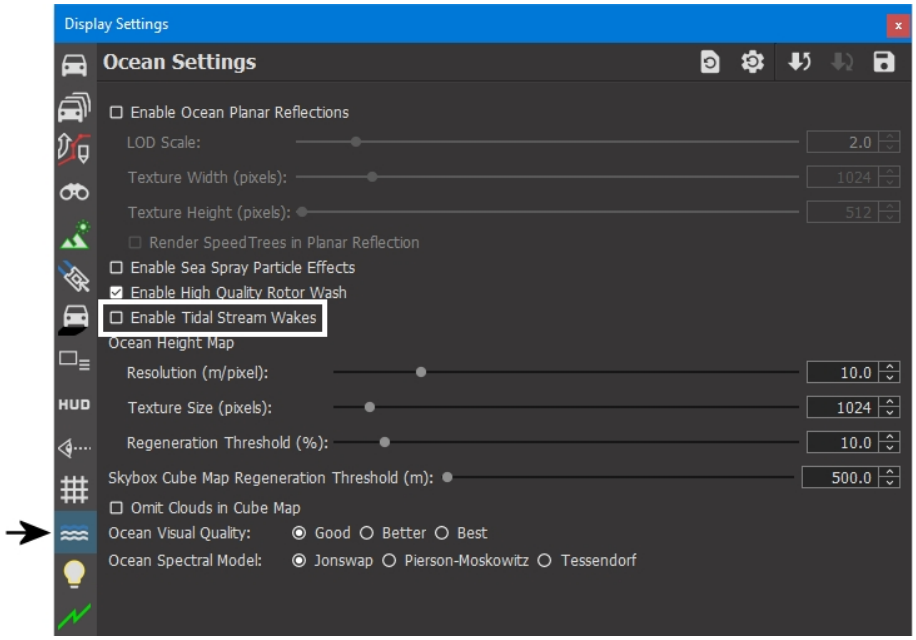
Tidal stream wakes are disabled by default.

> Even if tidal stream wakes are enabled, you will not see any wakes unless you also specify a tidal current, which is 0 by default. For details about setting tidal current, see "11.8. Marine Conditions" on page 341.

➤ **To enable or disable tidal stream wakes:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 126).

Figure 126. Ocean Settings



3. Select or clear Enable Tidal Stream Wakes.

11.8.4 Setting Ocean Quality

You can adjust the overall quality of visualization of the ocean. Increasing ocean quality decreases performance. Best quality may affect overall performance of your computer. We suggest using this setting primarily for demonstrations rather than large scenarios.

➤ **To set ocean quality:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 126 above).
3. Select one of the Ocean Visual Quality options.

11.8.5 Specifying an Ocean Wave Spectrum

In Ocean Settings, you can specify the wave spectrum to use for the ocean: JONSWAP, Pierson-Moskowitz, or Tessendorf. These are part of the Triton Ocean SDK. JONSWAP is the default spectrum.

The JONSWAP (Joint North Sea Wave Project) wave spectrum model is useful for maritime training applications and buoyancy in simulations where you require realistic ship motion. It is an extension of the Pierson-Moskowitz model, therefore using JONSWAP can have slower performance than Pierson-Moskowitz. Tessendorf is the least physically accurate, but it offers improved performance. For more information on each of these models, consult Sundog Software's documentation on the Triton Ocean SDK.

➤ **To set the ocean spectrum model:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 126 on the previous page).
3. Select one of the Ocean Spectral Model options.

11.8.6 Configuring Breaking Waves

VR-Forces can simulate the effect of waves breaking along the shore line. This feature uses shapefile data that defines the direction of the breaking waves. The *MAK Earth (online).mtf* terrain includes sample data for breaking waves. To see it, view the island of Oahu, Hawaii.

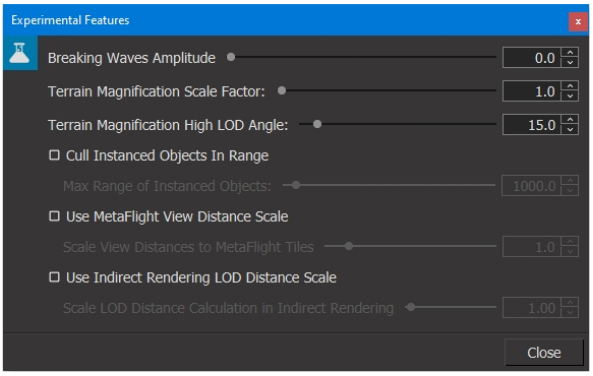
ⓘ You can set the amplitude of breaking waves. By default, the amplitude is 0.

This is considered an experimental feature. It may be useful, but may affect performance.

➤ **To configure breaking waves:**

1. Choose **Settings > Experimental Features**. The Experimental Features dialog box opens (Figure 127).

Figure 127. Experimental Features dialog box



2. Adjust the Breaking Waves Amplitude slider or type a value in the box. The maximum value is 10.0.

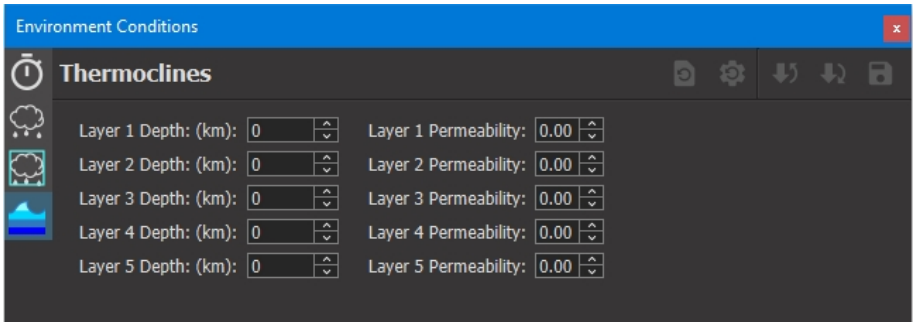
11.9. Setting the Thermocline

Thermoclines are layers of water that have different temperatures and, therefore, different density. Sound transmission varies as it passes through the thermocline. In VR-Forces, sonar effectiveness is reduced as it passes through thermocline boundaries. By default, VR-Forces does not set any thermoclines. You can specify the depth for up to five thermoclines.

➤ **To specify a thermocline:**

1. Choose **Simulation > Environment Conditions**. The Environment Conditions dialog box opens.
2. Select the Thermoclines page (Figure 128).

Figure 128. Thermoclines page



3. For each thermocline that you want to specify, enter or select a value in the depth box and the permeability box.

11.10. METOC Object Classes that Can Be Published

The published METOC object classes are based on both the global weather and any weather zones in the scenario. For details, see the *MAK ONE Products Interoperability Guide*.



12. Scenario Files

This chapter describes the files that are created when you create a scenario.

12.1. Introduction 350
12.2. The Scenario File 351
12.2.1 Scenario Parameters 351
12.2.2 Specifying Pathnames for Scenario Files 355
12.2.3 Changing the Terrain Database for a Scenario 356
12.3. The Plan File 356
12.4. The Object Map File 357
12.5. Editing the Scenario Objects File 357
12.5.1 Saving Default Parameters to the Scenario Objects File 358
12.6. The Scenario Extras File 358
12.7. Temporary Scenario Directories 358

12.1. Introduction

When you create and save a scenario, VR-Forces stores information about the scenario in these files:

- The scenario file, named *scenario_name.scn*, lists the locations of the other scenario files, the terrain database, and the simulation model set file, and it also sets several parameters for the scenario.
- The scenario objects file, named *scenario_name.oob*, stores identification and state information for simulation objects, control objects, and published tactical graphics.
- The object map file (*scenario_name.omp*) maps objects to the sim engines on which they were created.
- The plan file (*scenario_name.pln*) stores individual plans and global plans.
- The scenario extras file (*scenario_name.xtr*) lists the forces that each force is hostile to and it stores spawn pattern templates.
- The scripts file (*scenario_name.spt*) stores scenario-specific scripts.
- The overlay file (*scenario_name.ovl*) lists tactical graphics that are not published. Published tactical graphics are in the scenario objects file.
- The selection groups file (*scenario_name.sgr*) stores the selection groups in the scenario and their members.
- The saved views file (*scenario_name.osrx*) stores observer views, if any. It can also specify a startup view.
- The scenario-specific GUI settings file (*scenario_name.gui_settings*) contains GUI settings that only apply to this scenario.

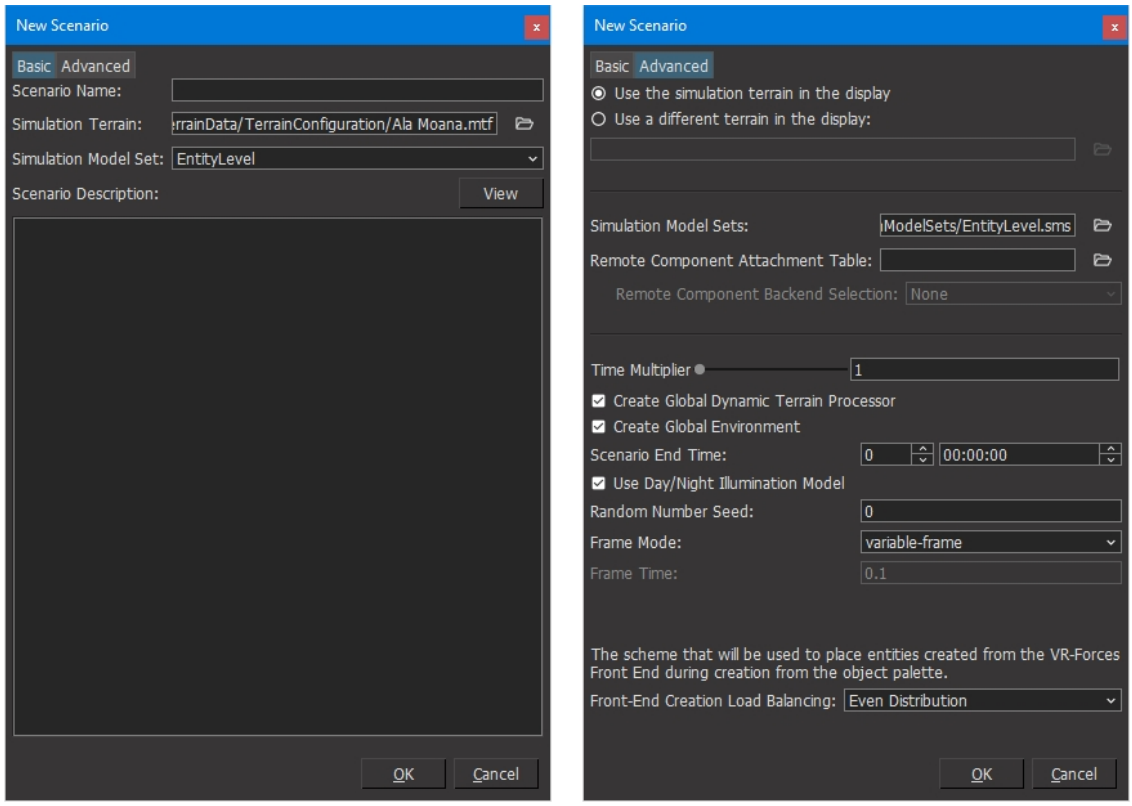
By default, scenarios are saved in a compressed zip archive with the extension *.scnx*. If you examine a compressed scenario file in an unzipping application, you will see that it contains the individual files described in the previous paragraphs.

For the most part, you will not want to edit scenario files by hand. In some cases, we recommend that you not edit them. However, this chapter explains how to edit them if you need to.

12.2. The Scenario File

When you create a scenario, you can set its parameters in the New Scenario dialog box (Figure 129).

Figure 129. New Scenario dialog box



The default values for scenario settings in the New Scenario dialog box are drawn from the file *./appData/settings/vrfGui/default-scenario.tmpl*. If you want to change the default settings, you can edit that file.

12.2.1 Scenario Parameters

A typical scenario file looks like the following (assume a scenario file called *myscenario.scn*):

```
(Scenario
  (Component-Attachment-Table  "")
  (Terrain-Database  "$(SHARED_DATA_DIR)\TerrainData\
      TerrainConfiguration\Ground_DB II.mtf")
  (Gui-Terrain-Database  "$(SHARED_DATA_DIR)\TerrainData\
      TerrainConfiguration\Ground_DB II.mtf")
  (Order-Of-Battle  "myscenario.oob")
  (Scenario-Scripts  "myscenario.spt")
  (Orbat "myscenario.orb"
  (Plan  "myscenario.pln")
  (Overlay  "myscenario.ovl")
  (SelectionGroups  "myscenario.sgr")
```

```
    (Object-Map  "myscenario.omp")
    (Scenario-extras  "myscenario.xtr")
    (time-multiplier  1.000000)
    (auto-reorganize False)
    (random-number-seed  0)
    (frame-mode  "variable-frame")
    (frame-time  0.100000)
    (run-duration-time 0.000000)
    (sim-time  0.000000) //no longer used
    (exercise-start-time  0.000000) // no longer used
    (additional-opds )
    (Simulation-Model-Set-Files "$(DATA_DIR)\simulationModelSets\
        EntityLevel.sms")
    (gui-runtime-scheme "DtRuntimeEvenDistributionLoadBalancer")
    (gui-runtime-scheme-data "")
    (remote-attachment-scheme "DtNoneBalancer")
    (remote-attachment-scheme-data "")
    (scenario-name  "My Scenario"
    (scenario-data
      (Create_Global_Dynamic_Terrain  "1")
      (Create_Global_Environment  "1")
      (DefaultObserverView  "1 - Absolute")
      (GuiObserverViews  "myscenario.osrx")
      (GuiScenarioSettings  "myscenario.gui_settings")
      (Language  "en")
      (ScenarioDescription  "Some text that describes the scenario.")
      (ScenarioExtentInformation  "-2.81258e+006,-4.33259e+006,
          3.72903e+006,938.957")
    )
    (version  3.200000)
)
```

Paths in the scenario files are relative to the application directory, for example, *./bin64*.

The Starting Time of Day is stored in the scenario objects file, not the scenario file.

Table 20 describes the parameters in a scenario file.

Table 20. Scenario file parameters

Parameter	Description
Component-Attachment	Specifies whether attachment of components to remote simulation objects is allowed in the scenario. The options are: • 0: no component attachment. • 1: the first sim engine manages all remote components. • 2: remote component attachment is load-balanced on all sim engines. Although you can set this value in the New Scenario and Load Scenario dialog boxes, it is not saved in the scenario file. To force a specific value, you must edit the scenario file.
Component-Attachment-Table	Specifies the component attachment table to use if attachment of components to remote simulation objects is allowed.
Terrain-Database	Specifies the terrain database to be loaded by the sim engine.
Gui-Terrain-Database	Specifies the terrain database loaded in the GUI.
Order-Of-Battle	Specifies the scenario objects file.
Scenario-Scripts	Specifies the file that has scenario-specific Lua scripts.
Orbat	Specifies the order of battle file.
Plan	Specifies the plan file.
Overlay	Specifies the overlay file.
SelectionGroups	Specifies the selection groups file.
Object-Map	Specifies the object mapping file.
Scenario-Extras	Specifies the extras file, which records force hostility relationships, and stores spawn pattern templates. It may be used for other purposes in the future.
time-multiplier	Specifies whether a scenario runs in real time or faster than real time. A value greater than 1 runs the simulation faster than real time. You can set this value dynamically using the Time Multiplier toolbar. If you are running a scenario using HLA Time Management, it is strongly recommended that you set **time-multiplier** to 1.
auto-reorganize	Allows you to specify the default behavior for unit reorganization. Set it to True or False. For information about reorganization, see "25.1. How Units Are Organized" on page 516.
random-number-seed	Sets the default random number seed. It provides a starting point for the generation of random numbers needed by VR-Forces. Enter an integer between 0 and the largest integer permitted on your system.

Table 20. Scenario file parameters (continued)

Parameter	Description
frame-mode	Sets the frame rate mode to one of: • Variable-Frame Run-To-Complete (variable-frame) • Fixed-Frame Best-Effort (fixed-frame) • Fixed-Frame Run-To-Complete (fixed-frame-run-to-complete). These frame rate modes are described in "3.4.3 Exercise Clock Modes" on page 122.
frame-time	Specifies the length of a frame, in seconds. If **frame-mode** is set to **fixed-frame** or **fixed-frame-run-to-complete**, you must set the frame time to a non-zero value. A value of zero for frame time prevents simulation time from advancing in either of these modes.
run-duration-time	Specifies how long the scenario should run before automatically stopping.
sim-time	This parameter is not supported in scenarios created after VR-Forces 4.0.3 and is only for backwards compatibility
exercise-start-time	This parameter is not supported in scenarios created after VR-Forces 4.0.3 and is only for backwards compatibility
additional-opds	Specifies custom OPD files to load (for backwards compatibility only).
Simulation-Model-Set-Files	Specifies a list of SMS files to load. The path is relative to the default data directory (*./data*) or a user-specified data directory.
gui-runtime-scheme	Internal name used to look up a code-based assignment scheme to determine on which sim engine to create new objects when they are created from the Create Object Palette.
gui-runtime-scheme-data	Data specific to the **gui-runtime-scheme** parameter.
remote-attachment-scheme	Internal name used to look up a code-based assignment scheme to determine which sim engine will host remotely attached entities.
remote-attachment-scheme-data	Data specific to the **remote-attachment-scheme** parameter.
scenario-name	An optional parameter that assigns a name to the scenario. It is independent of the file name and is only used in some message dialog boxes.

Table 20. Scenario file parameters (continued)

Parameter	Description
scenario-data	A block of data with these additional parameters: • **Create_Global_Dynamic_Terrain** specifies whether the global dynamic terrain processor should be enabled. If this is disabled, dynamic terrain does not work unless you create a dynamic terrain area. • **Create_Global_Environment** specifies whether the scenario supports environmental conditions. • **DefaultObserverView** specifies the observer view to use at startup. • **GuiObserverViews** specifies the saved views file for the scenario. • **GuiScenarioSettings** specifies the *.gui_settings* file. It records scenario-specific GUI settings. • **HiddenTerrainViews** lists views associated with the scenario terrain that are not displayed in the Observer Views Panel because they were cleared from the Observer Views Panel. • **ScenarioDescription** is the text of the scenario description. • **ScenarioExtentInformation** lists the extents of the scenario for the zoom to extents option. • **Set_Scenario_Start_Time_At_Local_On_First_Object_Creation** specifies the start time for the scenario, in seconds. • **UseDayNightModel** specifies whether to use the day/night illumination model.
version	Specifies versioning information for the scenario file. Do not edit this parameter.

12.2.2 Specifying Pathnames for Scenario Files

Pathnames in a scenario file can be absolute or relative. When VR-Forces saves a new scenario, it creates pathnames that are relative to the application directory. If you think you might need to move a scenario to another computer or a new directory structure, change the paths to be relative to the *./bin64* directory. To change pathnames, edit the scenario file in a text editor. VR-Forces preserves the new paths when it saves the scenario.

⚠ If you plan to run multiple GUIs and sim engines on multiple machines, particularly if they use different operating systems, you must change the pathnames for the terrain database and object parameter database to relative paths or simple pathnames.

Configuring How Paths are Interpreted

These parameters in *vrfSim.mtl* determine how pathnames for the object parameter database and terrain database are resolved:

- **defaultParameterDatabasePath** specifies the path to search for OPD files when an explicit path is not specified in a scenario file.
- **defaultTerrainDatabasePath** specifies the path to search for terrain database files when an explicit path is not specified in a scenario file.

When the sim engine loads a scenario file, it looks for the object parameter database using this search order:

1. If the path is absolute, load the file in the specified path.
2. If the path is relative, try to load the file relative to the application path (*./bin64/vrfSimprotocol*).
3. Look in the directory specified by **defaultParameterDatabasePath**.

When the sim engine loads a scenario file, it looks for the terrain database using this search order:

1. If the path is absolute, load the file in the specified path.
2. If the path is relative, try to load the file relative to the application path.
3. Look in the directory specified by **defaultTerrainDatabasePath**.
4. Use the path in the user-defined environmental variable VRF_TERRAIN_PATH, if it is specified.

12.2.3 Changing the Terrain Database for a Scenario

When you create a new scenario, you must choose terrain databases for the GUI and sim engine. By default, the sim engine uses the same database as the GUI.

You can edit the pathname of a terrain database in the scenario file or in the Scenario Properties dialog box. However, unless the new terrain database covers the same geographic area as the old one and is the same type of terrain (for example, UTM or geocentric), the scenario objects and overlay information associated with the scenario may be confusing or useless. (For details about editing Scenario Properties, see "7.4.3 Editing Scenario Properties" on page 251.)

12.3. The Plan File

VR-Forces saves all individual and global plans to a plan file (*.pln*).

Plan files are created interactively in the VR-Forces Plan window. The Plan window ensures that the plan is written using the correct syntax. We recommend that you do not try to create plan files or edit plan files by hand.

12.4. The Object Map File

When you start VR-Forces with multiple sim engines, you can specify which sim engines simulate the various objects you create. The object map file records the sim engines on which objects are simulated. The next time you open the scenario, VR-Forces looks for the sim engines specified in the object map file. If any of the specified sim engines is not available, the GUI asks if you want to remap objects to the sim engines that are present. You can run the simulation without remapping. However, the behavior of a simulation that is missing simulation objects is unpredictable.

Mappings between objects and sim engines are saved to the object map file. If you have a scenario that uses multiple sim engines and you are uncertain about the site:application values you need to use to run the scenario, you can view the object map file to refresh your memory. This example shows two entities and a route mapped to two different sim engines:

```
(map-entry
  (address  1 3201)
  (name  "M1A2 1")
)
(map-entry
  (address  2 3400)
  (name  "T80 2")
)
(map-entry
  (address  1 3201)
  (name  "route_1")
)
```

The **address** parameter lists the site and application number used for the sim engine on which the object was simulated.

If you want to run an existing scenario using sim engines with different site and application values, you could edit the object map file and change the address field to match the new site and application values. However, if you make any mistakes in this process, your scenario might not run correctly. As an alternative to editing the object map file by hand, you could use the load balancing option to remap objects to sim engines. For details about load balancing, see "7.5. Load Balancing Scenarios" on page 255.

12.5. Editing the Scenario Objects File

When you create a scenario, you populate it with simulation objects and tactical graphics. When you save or checkpoint a scenario, VR-Forces saves a record of each simulation object and tactical graphic, including its state information and geographic location, to the scenario objects file. The next time you load this scenario, VR-Forces creates the simulation objects using the settings in the object parameter database, but then overrides those settings with the information in the scenario objects file. Therefore, changes you make to the object parameter database might not be picked up by existing scenarios. If you want to change a basic value, you might need to edit the scenario objects file.

12.5.1 Saving Default Parameters to the Scenario Objects File

When a simulation object in a scenario is saved to the scenario objects file, parameters that are unchanged from the initial values in the object parameter database entry are written out in the form:

```
(parameter-name USE-DEFAULT)
```

The USE-DEFAULT keyword indicates that the next time this file is loaded (as part of a scenario) it should be initialized from data in the simulation object's OPD entry. The practical effect of the USE-DEFAULT keyword in scenario objects files is that if you modify the object parameter database, the next time you load a scenario, those changes can affect the simulation objects in the scenario.

This allows you to experiment with changes to parameters and view the effects of the changes to existing scenarios, without modifying the scenario objects file.

12.6. The Scenario Extras File

VR-Forces permits you to configure hostility relationships: for any force you can specify which forces it is hostile towards. By default, Friendly and Opposing forces are hostile to each other. If you change the hostility relationships, they are saved in the scenario extras file (*.xtr*).

This file also stores spawn patterns.

12.7. Temporary Scenario Directories

When you load a scenario, VR-Forces creates temporary directories in which it stores files that it requires to maintain and update the scenario. To prevent conflicts in the temporary directories when running multiple copies of VR-Forces from the same directory, VR-Forces creates unique temporary directories. The directories are created in the path specified by the TMP environment variable. If this variable is not specified on Linux, the */tmp* directory is used. The TMP variable must be specified on Windows. If VR-Forces cannot create a temporary directory, an error message is printed and the application exits. In the GUI, the temporary directory is created by *DtVrfRemoteController* and starts with the prefix *vrffe*. In the sim engine, the temporary directory is created by *DtCgfDispatcher* and has the prefix *vrfbe*. In both cases, the temporary directory is removed when VR-Forces shuts down.



III. Simulation Objects in Scenarios

The chapters in this section explain how to add simulation objects to a scenario, how to select them, and how to display information about them.



13. Introduction to Simulation Objects

Simulation objects are the entities and units that VR-Forces simulates as part of a scenario.

13.1. Simulation Objects 362
13.2. How Simulation Objects Are Identified 362
13.2.1 UUIDs 363
13.2.2 Simulation Object Names 363
13.2.3 Echelon IDs 363
13.2.4 Object IDs 364
13.2.5 Labels 364
13.3. How Simulation Objects Are Organized 364
13.3.1 How Echelon IDs Are Assigned 365
13.4. How Simulation Objects Communicate 365
13.4.1 Custom Communication Models 366
13.5. VR-Forces Uses a Component Architecture 366
13.5.1 Sensors 366
13.5.2 Controllers 367
13.5.3 Actuators 367
13.6. Simulation Object Behaviors and Tasks 367
13.7. Entity-Level Modeling and Aggregate-Level Modeling 368

13.1. Simulation Objects

The players in a simulation are called simulation objects. There are two broad classes of simulation objects:

- Entities represent a discrete individual vehicle or person, such as a tank or a submarine.
- Units represent a hierarchical group of entities, units, or both, such as a platoon or a company.

The concepts described in this section are generally applicable to entity-level modeling and aggregate-level modeling. For details about the differences between the two, see "13.7. Entity-Level Modeling and Aggregate-Level Modeling" on page 368.

Simulation objects can act in and respond to their environment. They execute tasks and they have behaviors. Each simulation object is part of a force and may be a member of an organizational unit.

The VR-Forces application comes with a set of simulation object models that you can instantiate in a simulation. Application developers can create additional simulation objects with the VR-Forces toolkit. You can also create new simulation object types in the Simulation Object Editor. For information about creating new simulation objects, see "Section XIII. Creating and Editing Simulation Models" on page 1301.

13.2. How Simulation Objects Are Identified

VR-Forces identifies simulation objects by UUID (universal unique ID), name, echelon ID, object ID, and label. Table 21 summarizes the characteristics of each type of identifier.

Table 21. Simulation object identifiers

Identifier	Characteristics
UUID	• Unique. Assigned by VR-Forces. • Persists from exercise to exercise.
Name	• Persists from exercise to exercise. • Set by user. • Limits on the length of the character string. • Does not have to be unique.
Echelon ID	• Must be unique. • Persists from exercise to exercise. • Assigned by VR-Forces. Changes if an entity is organized into a unit.

Table 21. Simulation object identifiers (continued)

Identifier	Characteristics
Object ID	• Unique. • Does not persist from exercise to exercise. • No user control.
Label	• Does not have to be unique. • Persists from exercise to exercise. • Controlled by user. • No limit on character length.

13.2.1 UUIDs

UUIDs are unique identifiers generated by VR-Forces to ensure that every simulation object can be uniquely identified and that the identification persists each time you run the scenario. In particular, the use of UUIDs supports importation of scenarios because while simulation objects in different scenarios might have the same names, they will not have the same UUIDs, so it is possible to import them and not change their names if you do not want to. (The default is to rename duplicate simulation objects.)

13.2.2 Simulation Object Names

A name is a text string that, by default, consists of the simulation object's short name, as defined in the object parameter database, and an integer. For example, if you create a new scenario and add two M1A2 tanks and two Apache helicopters, the default names are **M1A2 1**, **M1A2 2**, **AH-64A 1**, and **AH-64A 2**. When VR-Forces generates a name, it picks an integer that makes the name unique. However, if you import a scenario and do not specify that duplicate names be renamed, it is possible that more than one simulation object will have the same name.

You can edit a simulation object name to make it meaningful to you.

Simulation object names have restrictions on their length:

- Entities – 11 characters.
- Units – 31 characters.

If you are using a character set that requires multiple ASCII characters to represent a visible character, entity and unit names will need to be correspondingly shorter.

13.2.3 Echelon IDs

An echelon ID is a text string that identifies the simulation object's place in the force hierarchy. For example, if you create an **M1A2 1** in Blue force (or Force 1), the echelon ID would be **1 M1A2, 1 Force**. The syntax of echelon IDs and the details of simulation object

organization are described in "13.3. How Simulation Objects Are Organized" below. You cannot edit a simulation object's echelon ID. The echelon ID changes if you change the force hierarchy by adding, removing, or editing simulation objects.

13.2.4 Object IDs

Each simulation object has a unique object ID (global object designator). The global object designator is equivalent to the HLA object name in HLA and the DIS entity identifier in DIS. The object ID is used to identify the simulation object internally and for network communications.

13.2.5 Labels

A label is a text string that you can use to identify simulation objects without the restrictions of uniqueness or character length that affect the other identifiers.

You can also identify simulation objects using extended labels – additional text strings that you can give descriptive names to. For details, see "21.3. Using Extended Labels" on page 477.

> In the VR-Forces GUI, entity labels refer to informational graphics displayed next to simulation objects. The label described in this section is one of the symbol decorations displayed in Plan View observer mode as part of the entity label. For more information about the different types of labels and information displays, see "21.2. Displaying Entity Labels" on page 469 .

13.3. How Simulation Objects Are Organized

All simulation objects simulated by VR-Forces exist in the context of a hierarchy. At the top-most level, simulation objects are grouped according to force ID (friendly, opposing, neutral, and so on). At lower levels, simulation objects are grouped according to military organization, in which a simulation object has a designation in the hierarchy, such as 2nd Platoon, B Company. At each echelon level in the organization, a simulation object has a single superior simulation object. Units contain one or more subordinate entities, units, or both. The hierarchy's lowest level contains an individual entity.

Each simulation object has an echelon ID in the format:

```
designator [category] [echelon], designator [category] [echelon],
  ..., force_number Force
```

where:

- *designator* is a number or letter that differentiates the simulation object from other members of the echelon, such as 1 (M1A1), 2 (M1A2), and so on.
- *category* is an optional descriptor for the simulation object, such as M1A2.
- *echelon* is an optional descriptor for the organizational unit, such as "Platoon".
- *force_number* is the force the simulation object belongs to, based on the force type enumerations. The force is the highest level in the hierarchy.

When you create a new simulation object, it exists only within the default force. Therefore, its echelon ID would be similar to:

```
1 M1A2, 1 Force
```

If a simulation object becomes part of a unit, the unit information becomes part of the echelon ID, and the designator may change, for example:

```
2 M1A2, 2 Plt, 1 Force
```

13.3.1 How Echelon IDs Are Assigned

The short name and echelon labels that make up an echelon ID, such as "M1A2", "Plt", and so on, are specified in the VR-Forces object parameter database and are associated with object-types. When you create a simulation object of a certain type, the short name and echelon name associated with that type are assigned to the simulation object. You cannot change the labels interactively. You can change the labels by editing the object parameter database.

Designators are assigned by VR-Forces. You have no direct control over them. When you create new simulation objects, they are assigned designators sequentially, starting with one (1). If you aggregate a group of entities, the unit leader is assigned the designator 1, and the other entities in the unit are assigned successive designator numbers.

13.4. How Simulation Objects Communicate

VR-Forces includes a default communication model called simple-radio-comm-model. Using this simple communication model, radio messages always reach their intended destination (as long as the destination is on a reachable network), and messages are all passed instantly. The simple radio communication model has one parameter, which determines how the radio networks are set up. There are three connection-modes available:

- **aggregate** mode specifies that each unit has its own radio network. Messages sent from a simulation object can reach any simulation object that is in the same organization, at the same level (the simulation object's peers in the organization), and can also reach the simulation object's direct superior. All other simulation objects are outside the network and are unable to receive messages sent by the simulation object. This is the default mode.
- **force** mode specifies that there is a radio network for each force in VR-Forces. Messages sent from a simulation object can reach any other simulation object of the same force.
- **all** mode specifies that all simulation objects in VR-Forces are part of one large radio network. In this case, any simulation object can receive messages sent by any other simulation object.

The communication model is configured in *./data/simulationModelSets/<sms>/commModelParams.mtl*. For details about configuring radios, see "80.2. Configuring Radios" on page 1574.

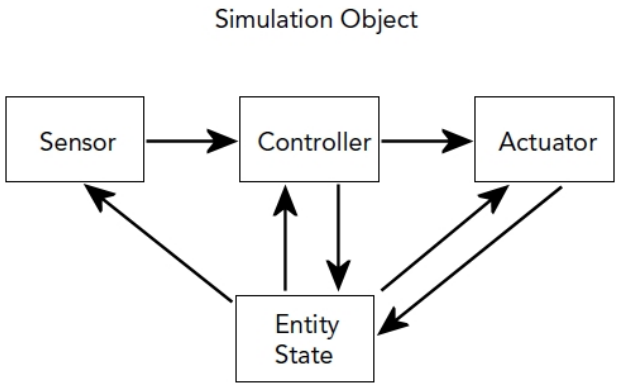
13.4.1 Custom Communication Models

You can create new communication models and install them in VR-Forces. This is useful if you want to define your own connectivity model or write your own algorithms for determining whether radio messages are delivered. For details, please see the API documentation.

13.5. VR-Forces Uses a Component Architecture

VR-Forces simulation objects use a component architecture similar to that used in many robotics applications. There are three basic types of components supported: sensors, controllers, and actuators. Components communicate with each other through data ports and port groups, which can provide data as simple as a single number (such as a throttle value) or more complex data such as a list of simulation objects that have been detected. The port groups make it possible to multiplex and prioritize multiple data sources to a single data recipient. For example, the automotive actuator of ground vehicles can receive inputs from a collision-avoidance controller, move-to-point controller, and follow-route controller, but it does not want inputs from all of those controllers at once.

Figure 130. Simulation object component architecture



The component architecture supports any number of sensors, controllers, and actuators in a simulation object. Components are specified in the object parameter database by using component descriptors. Component descriptors indicate the type of component that should be created. They can also provide parameter information to a component.

For more information, see "Appendix D. Object Parameters" on page 1681.

13.5.1 Sensors

Sensor components provide models of the simulated environment, which are used by controller components to make decisions and perform tasks. The simplest sensor might provide (simulated) ground truth, while more sophisticated ones could use complex models for sensing electromagnetic emissions, RADAR, or a cognitive model to simulate a soldier or crew member's perception of the simulated world. Sensor components can get information from:

- The virtual network.
- From a terrain database.
- By monitoring the state of the simulation object model.
- Many other potential sources.

The use of sensors is optional.

13.5.2 Controllers

Controller components typically provide control inputs to actuator components (although this is not required). Controllers can use information provided by sensors to perform their functions, but this is not a requirement either. For example, the controller that implements the various forms of the Wait task neither requires sensor information nor provides control inputs to any actuators.

13.5.3 Actuators

Actuator components:

- Provide the physical model of the simulation object being simulated.
- Change the simulation object's state.
- Interact with other simulation objects (such as by sending messages).

Actuators can use control inputs provided by controller components as parameters to its model each simulation frame. Actuators may:

- Send messages on the radio system.
- Generate events on the virtual network, such as detonations.
- Modify the state of the simulation object: position, velocity, damage, and so on.

Actuators can also use data directly from a sensor, when modeled or filtered data is preferred over simulated ground truth for a model that does not require a controller.

13.6. Simulation Object Behaviors and Tasks

A behavior is something that a simulation object knows how to do on its own. Behaviors describe how a simulation object reacts to a specific condition or situation or how it performs a task. A task is something that you tell a simulation object to do. A simulation object does not innately know that it should move to a point. However, if you task it to move to a point, it knows how to locate the point and how to move to it.

VR-Forces has behavioral models for the included simulation object types. These models simulate the behaviors you would expect from live participants and objects functioning in the real world. You can change the parameters for the behavior models. With the VR-Forces toolkit, you can change the models themselves and add additional models. For more information, please see the *VR-Forces Developers Guide*.

- No semi-automated forces (SAF) system, such as VR-Forces, can completely emulate the behavior of a complex real-world entity. The behaviors provided by VR-Forces handle the lower-level activities necessary to complete a task and allow you to focus on the higher level activities involved in assigning tasks and preparing plans.
- If you do not want a simulation object to engage in any independent intelligent behaviors, you can disable them with the Autonomous Actions Enabled set data request. For details, see "40.3. Autonomous Actions Enabled" on page 866.
- You can also write behaviors for units. For details, see "Chapter 38. Writing Advanced Scripts with the Behavior Engine" on page 815.

13.7. Entity-Level Modeling and Aggregate-Level Modeling

VR-Forces supports two different approaches to modeling simulation objects: entity-level simulation and aggregate-level simulation.

In entity-level simulations, VR-Forces models the interaction among individual entities, such as ships, tanks, automobiles, airplanes, and people. When you create a simulation object, you are usually creating one tank, ship, human, and so on. Combat takes place between entities based on their weapon systems and their ability to withstand munitions.

In an entity-level simulation, you can combine entities into higher echelon units, such as platoons and companies, but when they engage in combat, they do so by modeling the interactions of their members at the individual entity level.

Aggregate-level simulations focus on the high-level movement and interaction of large numbers of personnel and equipment in hierarchical units. They use abstractions when modeling forces rather than modeling individual simulation object interactions and combat. When you create a simulation object, you are usually creating a company, a brigade, or another object representing an entire group. Movement, combat, and other capabilities are abstractions of entity-level models. Aggregate-level combat is modeled on the relative strengths and weaknesses of simulation objects in personnel, equipment, and location, not on the engagement of individual entities. For example, attrition equations are used in combat instead of individual weapon engagement models.

In both entity- and aggregate-level simulations, you can combine entities and units into higher echelon units.

The type of modeling used in a scenario depends on the systems that simulation objects use, which are configured in the simulation model set it uses. If you use *EntityLevel.sms* or an SMS similar to it, it uses entity-level modeling. If you use *AggregateTacticalLevel.sms*, or an SMS similar to it, it uses aggregate-level modeling. Therefore, when you create a scenario, you must consider which type of modeling you want to use and you must remember which type of modeling you are using when you populate the scenario.

Aggregate-level modeling only works with HLA Evolved, HLA 4, and MAK FOM extensions. (If you try to create a scenario with *AggregateTacticalLevel.sms* and you are using an unsupported protocol, VR-Forces displays an error message.) For a detailed discussion of the aggregate warfare model, see "Chapter 27. Aggregate-Level Modeling" on page 527.

Aggregate-level SMSs can have simulation objects that represent individual entities. However, they use data-driven models that are different from those used by similar entities in *EntityLevel.sms*.

Although it is technically possible to create a scenario that uses both types of simulation model sets, the resulting behavior will be unpredictable because simulation objects that use aggregate-level modeling cannot interact with simulation objects that use entity-level modeling. Combining these SMSs is not supported and not recommended.



14. Creating and Placing Objects

This chapter describes generic procedures for creating and placing simulation objects, tactical graphics, and simulation effect objects. The later chapters assume that you are familiar with these procedures.

14.1. Creating Objects 372
14.2. Selecting the Object to Create 373
14.2.1 Selecting the Object from the Create Object Palette 374
14.2.2 Selecting the Object to Create on the Create Menu 377
14.2.3 Filtering the Create Object Palette 377
14.2.4 Previewing Object Information 380
14.2.5 Draw Mode 381
14.3. Placing Objects 382
14.3.1 Placing an Object Using Default Values (Click to Create) 382
14.3.2 Specifying an Object's Properties Before You Create It (Click to Locate) 384
14.3.3 Default Placement of New Entities 386
14.3.4 Placing Entities Inside Buildings 386
14.3.5 Placing Simulation Objects on Other Simulation Objects 386
14.4. Specifying an Object's Altitude 386
14.4.1 Setting Altitude Dynamically 387
14.4.2 Setting Altitude in the Create Object or Edit Object Dialog Box 387
14.4.3 Specifying the Altitude for All of the Vertices in a Route 388
14.5. Specifying an Object's Heading Dynamically 389
14.6. Specifying an Object's Pitch Dynamically 389
14.7. Locking the Mouse to the Object Being Created 390
14.8. Simulation Object Resources 391
14.9. Moving Objects 391
14.9.1 Dragging an Object to a New Location 392
14.10. Editing Simulation Objects 393
14.10.1 Editing an Object's Name in the Object List 394

14.11. Copying and Pasting Objects 395
14.11.1 Copying Objects 396
14.11.2 Pasting Objects 396
14.11.3 Pasting Specific Entity Characteristics 397
14.12. Deleting Simulation Objects 399
14.13. Reinitializing Simulation Objects 399
14.14. Simulation Object Groups 400
14.15. Creating Random Versions of Similar Entity Types 400
14.16. Creating Cultural Features 400
14.16.1 Embedding Cultural Features in the Terrain 401
14.17. Creating Props 402
14.18. Adding an Object to the Favorites List 402
14.18.1 Removing an Object from the Favorites List 403

14.1. Creating Objects

VR-Forces lets you create entities, tactical graphics, simulation effect objects (terrain, weather, clutter, and so on), cultural features, and props. You can create most of these from the Create menu or the Create Object Palette with the exception of props, which you create only from the Props Palette. The procedure for creating an object is essentially the same regardless of the type of object:

1. Select the object that you want to create.
2. Specify the location and properties of the object.

However, there are several ways to select the object you want to create and several ways to specify an object's location and properties. This chapter describes the different generic ways to create objects. Additional procedures specific to different object types are described in later chapters.

When you create an object, VR-Forces adds a tab to the right side of the window, below the object creation palette(s). The tab is labeled Create *object*, where *object* is whatever object you selected to create. If you click that tab, it opens a dialog box where you can specify the properties of the object you are creating. Table 22 lists the properties that you can set for each object type. You can also edit these same properties after you create an object.

Table 22. Editable object properties

	Entity	Tactical Graphic	Cultural Feature	Simulation Effect Object	Prop
Altitude	X (specific entities)	X		X (specific objects)	
Force	X	X	X	X (specific objects)	
Heading	X		X		
Label (and extended labels)	X	X	X	X	
Location	X	X	X	X	X
Name	X	X	X	X	
Overlay	X (Edit only)	X	X (Edit only)	X	
Pitch			X (specific objects)		
Publish Object		X		X (specific objects)	
Style		X			
Visual Markings	X (specific entities)				

14.2. Selecting the Object to Create

You can select the object to create on the Create menu or on an object palette (Create Object Palette or Props Palette). Once you have selected the object, placing the object is the same regardless of how you started the process. For details, see "14.3. Placing Objects" on page 382.

The Create menu does not contain the full list of objects configured in VR-Forces. It only lists the simulation objects and tactical graphics that you have marked as "Favorites". Marking favorite objects allows you to quickly access those objects you use most frequently. For details, see "14.18. Adding an Object to the Favorites List" on page 402.

14.2.1 Selecting the Object from the Create Object Palette

VR-Forces has palettes that you can use to quickly choose the object that you want to create:

- Create Object Palette. You can create many different objects from this palette:
  - Simulation objects. Entities and cultural features
  - Tactical graphics. Control objects, lines, circles, points, text, volumetric lines and areas.
  - Simulation effect objects. Terrain, weather objects, clutter areas, various types of hazards, obstacles, and pedestrian-oriented objects. (Some objects are aggregate-level only.)
- Props Palette. Create props that can be saved as part of the terrain. (The default window layout does not display the Props Palette. You can open it from the View menu.)

Figure 131 shows the location of the palettes (with the Props Palette enabled).

Figure 131. Object creation palette tabs

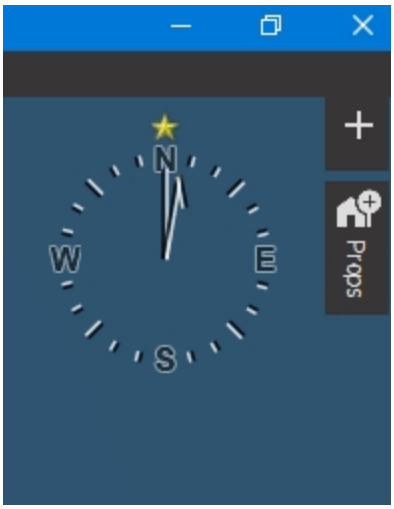
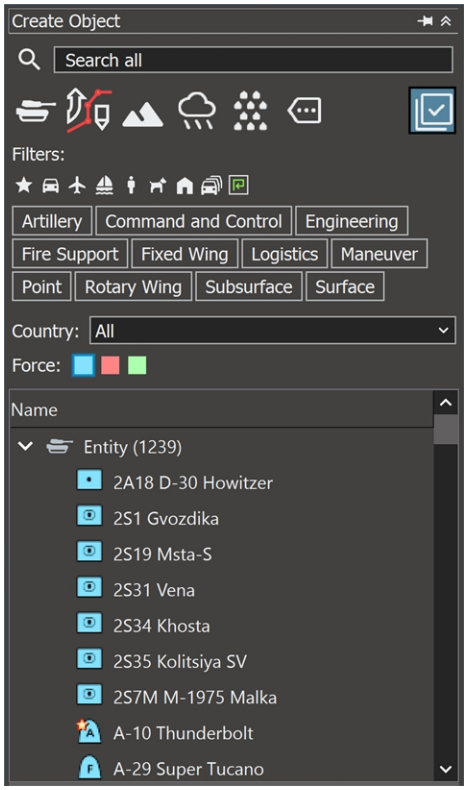


Figure 132 shows the Create Object Palette when it is open.

Figure 132. Create Object Palette



The objects in the palette are identified by their 2D icon and their name. You can use images other than the default icons to identify objects. For details, see "Using an Alternative Icon List" on page 1344.

➤ **To select the object to create on the Create Object Palette:**

1. If you are using multiple simulation engines, in the Selected Simulation Engine Toolbar select the simulation engine on which you want to simulate the object. You can also use Automatic (the default option) to have VR-Forces add the simulation object to the least burdened sim engine.
2. Click the tab to open the palette (Figure 132).

   The palette lists all objects that can be created. There is no preconceived force for any of them. You can filter the palette list by type of object (entity, tactical graphic, terrain, weather, clutter, or simulation object), text string, tag, and country of use. (The Props Palette organizes props in a tree list.)
3. On the Force line, click the button that represents the force for which you want to create this object.
4. Optionally, filter the list of objects. (For information about how to filter object creation palettes, see "14.2.3 Filtering the Create Object Palette" on page 377.)
5. Optionally, filter the list of entities by selecting a country in the Country list.

> The Country filter lists simulation objects of the selected tag that are used by the specified country. VR-Forces does not provide an exhaustive list of countries or the platforms that they use. You can add additional countries and categorize simulation objects in the Simulation Object Editor. For details, see "69.1.9 Editing a Simulation Object's Country List" on page 1342.

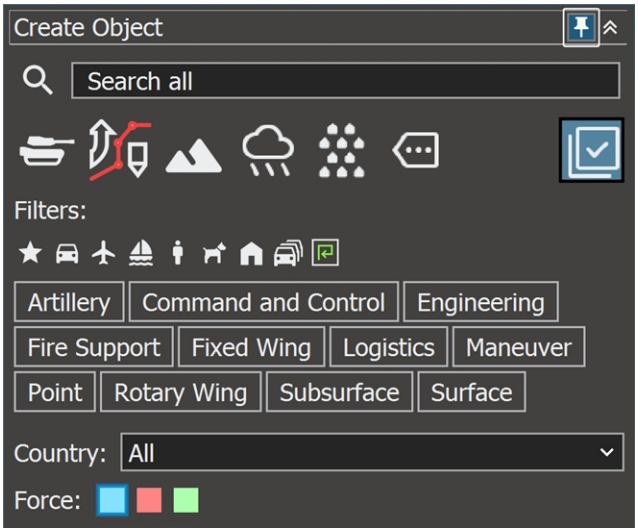
6. Optionally, enter a search string to further filter the list.
7. In the list of objects, select the object that you want to create. Information about that object displays at the bottom of the palette. The cursor changes to draw mode and a Create *object* tab is added to the window. For details about draw mode, see "14.2.5 Draw Mode" on page 381.
8. Create the object, as described in "14.3. Placing Objects" on page 382.

Pinning the Object Palette to the Window

Ordinarily, after you select an object to create, the palette closes. However, you can pin the palette to keep it open so that you can select additional objects to create without needing to click the palette to reopen it. If you pin a palette, it becomes dockable and undockable and you can drag it off the window like the other panels. (For details about undocking panels, see "84.1. Dockable Panels and Toolbars" on page 1622.)

➤ To pin a palette so that it stays open after you select an object, click the pin button (📌) in the palette's title bar (Figure 133).

Figure 133. Palette pin



14.2.2 Selecting the Object to Create on the Create Menu

Waypoints, routes, areas, and objects marked as Favorites are available on the Create menu.

➤ **To select the object to create on the Create menu:**

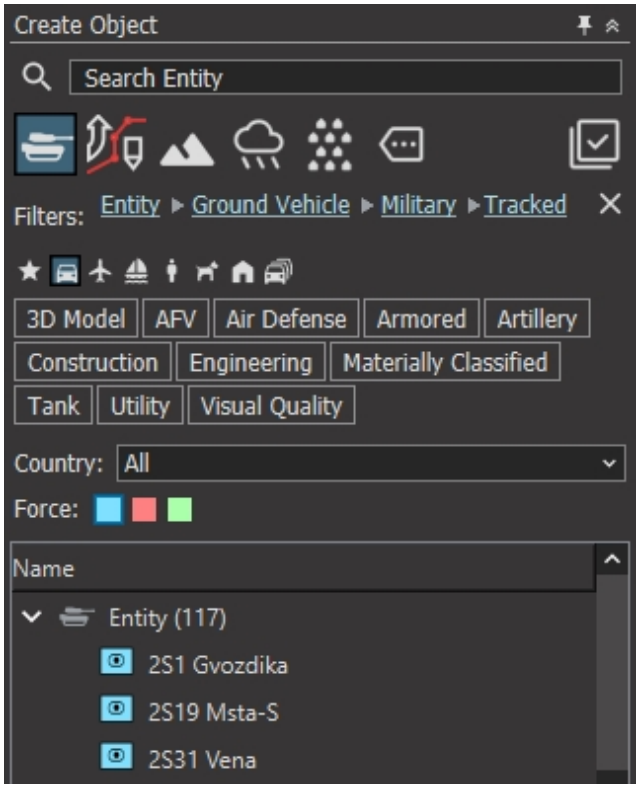
1. If you are using multiple simulation engines, in the Selected Simulation Engine Toolbar, select the simulation engine on which you want to simulate the object. You can also use Automatic (the default option) to have VR-Forces add the simulation object to the least burdened sim engine.
2. To create an object or graphic, choose **Create >** *type* **>** *object type*, where:
   - *type* is a category of object, such as Entity or Tactical Graphic.
   - *object type* is a specific object within the category, such as M1A2 SEP Abrams or Weather (Area).

   The cursor changes to draw mode and a Create *object* tab is added to the window. For details about draw mode, see "14.2.5 Draw Mode" on page 381.
3. Create the object, as described in "14.3. Placing Objects" on page 382.

14.2.3 Filtering the Create Object Palette

You can filter the Create Object Palette to make it easier to find the object that you want to create. You can filter objects by type, tags, country of use, and text strings.

Figure 134. Create Object palette filtering



Types are the kinds of objects you can create: entities, tactical graphics, and simulation effect objects: terrain, weather, clutter, and others. When you select a type, the Tag window updates to show only the tags that apply to objects of that type. The object list is also filtered, listing only objects of that type.

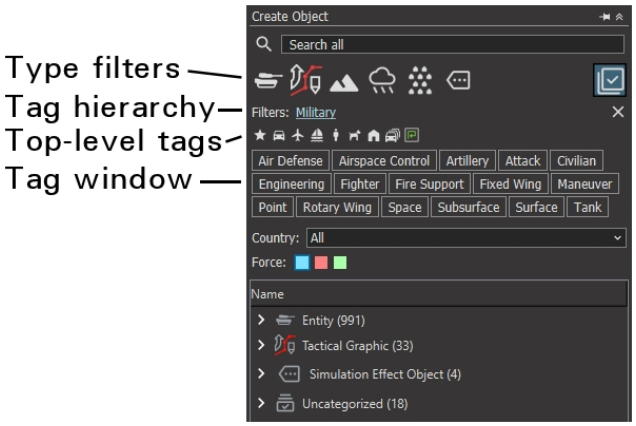
Tags are categories, such as land, air, military, civilian, and so on.

Many object types are used by multiple countries. If you want to see the objects used by a particular country, you can select the country of interest. (This filter does not apply to tactical graphics.)

Using the Search field, you can enter a text string to search for object names. As you type, objects that contain that string are listed. The Tag window updates to show only the tags that apply to the listed objects.

Simulation objects are assigned to tags and countries in the Simulation Object Editor. For details, see "Chapter 68. The Simulation Object Editor and SMSs" on page 1303 and "Chapter 69. Editing Common Parameters" on page 1335.

Figure 135. Create Object Palette filters



➤ **To filter the objects listed in the Create Object palette:**

1. To filter by type, select the type you want to filter by. For details, see "Filtering the Create Object Palette by Type" below.
2. To filter by tags, select the tags you want to filter by. For details, see "Filtering the Create Object Palette Using Tags" on the facing page.
3. To filter by country, select a country in the Country list.
4. To filter by text, enter a text string in the Search box.

Filtering the Create Object Palette by Type

All objects have a particular type:

- Entity (). Ground vehicles, aircraft, surface and subsurface vehicles, humans, animals, cultural features, and units.
- Tactical Graphic (). Lines, circles, points, text, volumetric lines and areas, and some standard military tactical graphics.

- Simulation effect objects. These are objects that can affect the simulation. There are four types of simulation effect objects:
  - Terrain (). These objects make changes to the terrain. Some can change the appearance of the ground in dynamic terrain, while others can affect how human characters navigate the terrain.
  - Weather (). Local weather zones.
  - Clutter (). Areas and points that simulate pattern of life.
  - Simulation Effect Object (). Other simulation effect objects that are not one of the other three types.

If you want to search for simulation objects that are not in a category, right-click the type button and choose Not. The type displays with a not icon on the Filters line.

To clear the type filter, click All ().

Filtering the Create Object Palette Using Tags

All objects are categorized using tags. An object can have multiple tags. For example, a ground vehicle can have the Ground Vehicle tag, but it can also have other tags such as Military, Tracked, Materially Classified, and so on. You can filter for any of these tags.

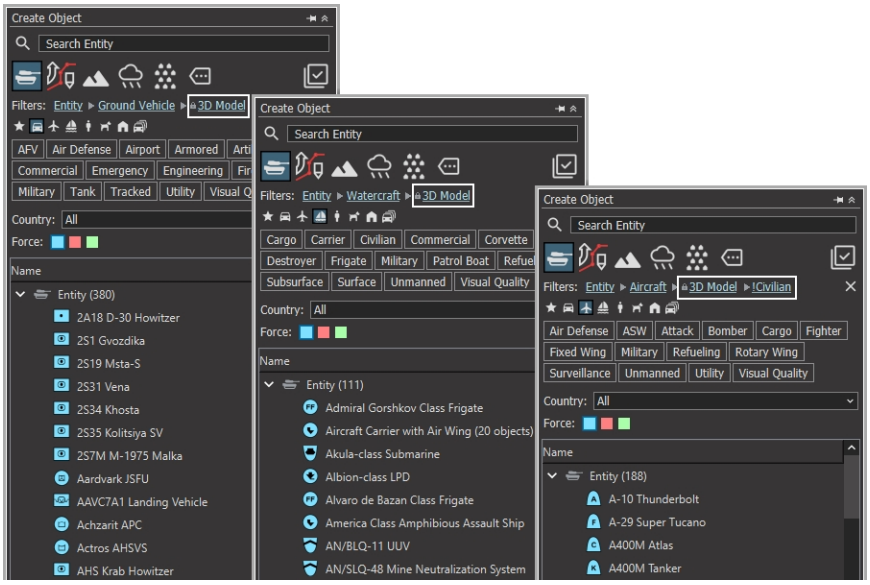
Tags are listed in the Filters section of an object creation palette. The top row of tags (represented by buttons) is static during any VR-Forces session (Figure 135). It contains the highest ranking tags. As you select tags, the Filters: line lists the tags you have selected from most inclusive to least inclusive. You can clear any level of filtering by clicking the label to its left.

The Tag window lists the tags that apply to simulation objects in the objects list that have not yet been filtered out. This list redisplays as you select additional tags. (If the number of tags that could be listed in the Tag window exceeds the space available, VR-Forces displays tags based on frequency of use and ranking. For details about ranking tags, see "68.11.1 Adding a New Tag" on page 1327.)

VR-Forces remembers the tags selected for each top level tag. So if you set up some tags for Ground Vehicle and then filter using the Watercraft tag, if you go back to the Ground Vehicle tag, the tags you selected previously are still selected.

You can make tags "sticky". A sticky tag applies to all top level tags. For example, suppose you only want to see simulation objects that have 3D models, regardless of the platform. You could make 3D Models a sticky tag. Then no matter which top level tag you selected, it would be filtered for the 3D Models tag. Sticky tags displays with a lock icon on the Filters line (Figure 136). You can also set a filter to look for simulation objects that are not in a category. Not tags display with a no circle icon on the Filters line.

Figure 136. Sticky and Not tags



➤ **To filter a list using tags:**

1. Click a top level tag, or All.
2. Select additional tags in the Tag window to narrow the filtering. As you select tags, the object list redisplays and the tag list redisplays to show the tags that apply to the remaining members of the object list. As you select additional tags, the hierarchy of tags on the Filters line updates.
3. If you always want to use a certain tag, such as 3D Model, or Materially Classified, right click the tag button and choose Sticky. The tag is displayed with a lock icon on the Filters line. It applies to all top level tags.
4. If you want to search for simulation objects that are not in a category, right-click the tag button and choose Not. The tag is displayed with a not icon on the Filters line. It applies to all top-level tags.
5. To clear tags, in the breadcrumb path on the Filters line, click the tag to the left of the tag you want to clear. To clear all tags, click the X button.

14.2.4 Previewing Object Information

The Create Object Palette provides a preview of the selected object.

- Entities display an image of the 3D model or, if there is no 3D model, the 2D icon. It also shows the tags defined for that entity.
- Tactical graphics display the tags, and some also display a summary of the tactical graphic's functionality.
- Simulation effect objects (terrain, weather, clutter, and so on) display a summary of the effect, and any tags defined for the object.

If you click one of the tags for the selected object, that is immediately applied as a filter to the objects list in the Create Object Palette.

14.2.5 Draw Mode

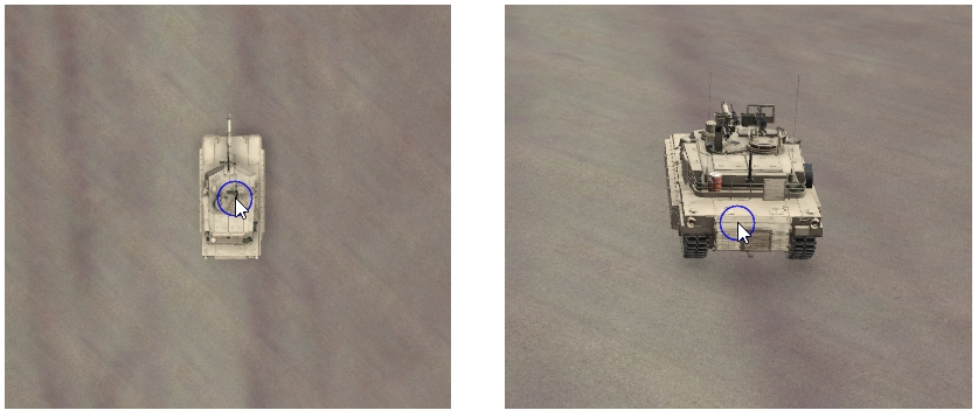
When you create an object, the cursor changes to draw mode. If you are creating a simulation object or a cultural feature in the 2D view, the cursor changes to a 2D icon that represents the chosen simulation object (Figure 137).

Figure 137. 2D entity creation and Create *object* tab



In the 3D view, the cursor changes to the 3D model for the chosen entity (Figure 138). If you are zoomed out, 3D models may be hard to see. If you are creating multiple human characters of a type that randomizes the appearance of the character that will be created, such as Civilian Male, after each character you create, the model attached to the cursor changes to show you the appearance of the next character appearance that was randomly selected.

Figure 138. 3D entity creation



Top-down view | Ground-level view

When you create a tactical graphic, the cursor displays the waypoint symbol when you create a point and a vertex symbol when you create any type of line or area. Figure 139 illustrates tactical graphics draw mode cursors.

Figure 139. Tactical graphics cursors in draw mode



When you create a prop, the cursor changes to show the model for the prop. In the 2D view, the model is a top down view and is proportional to the terrain, so it might be difficult to see if you are zoomed out (depending on the size of the prop).

14.3. Placing Objects

After you select the object that you want to create, the Create *object* tab is added to the right side of the window below the palette(s).

There are two basic methods for placing simulation objects:

- Click to Create. Each mouse click creates an instance of the selected object. For multi-vertex tactical graphics, each click specifies a vertex. Click to Create is best for rapidly creating objects using the default values for the object's properties.

  The create object graphic is attached to the mouse cursor. Although you can use the Create *object* dialog box to specify an object's name or other properties, as you move the cursor into the dialog box, the creation location (as represented by the object graphic) moves with it, which can be disconcerting. You can set heading and altitude dynamically, so there is no need to use the Create *object* dialog box for these properties.

- Click to Locate. Mouse clicks specify the coordinates, but do not cause the simulation object to be created. The create object graphic is placed at the specified coordinates, but is not attached to the mouse cursor.

  If you want to explicitly set the object's location, and particularly if you want to set the altitude relative to sea level, you must use this method. This method may also be more comfortable for persons who want to take a deliberative approach to specifying an object's properties before creating it.

Once you have clicked to place a simulation object, an icon indicates the location where it will be created as a placeholder. This tells you that your click was successful while VR-Forces creates the object. The indicator at the bottom right of the main window also displays the process of adding the object.

14.3.1 Placing an Object Using Default Values (Click to Create)

Although you can specify an object's properties using the Click to Create method, it is best suited for quickly creating objects using default properties. Objects are placed on the default overlay for the object type.

You can use this method to automatically embark simulation objects and attach tactical graphics. For details, see "15.1. Embarking and Disembarking Simulation Objects" on page 406 and "47.8.1 Attaching a Tactical Graphic Automatically" on page 1029.

➤ **To place an object with each mouse click:**

1. Select the object that you want to create as described in "14.2. Selecting the Object to Create" on page 373. A Create *object* tab is added to the side of the window below the palettes.
2. For simulation objects and waypoints, click on the terrain for each object you want to place. You can create multiple single-point objects without exiting create mode. For details about disabling this create mode, see "14.7. Locking the Mouse to the Object Being Created" on page 390.

   For multi-vertex tactical graphics, click on the terrain to place each vertex. Double-click to place the last vertex in a route or area. (Do not try to close areas by clicking on the starting point. Doing so only creates another vertex near the first one.)

   To draw a circles, ellipses, boxes, or text, see the sections in "Chapter 47. Creating and Editing Tactical Graphics and Control Objects" on page 1011.

   - If new objects or vertices are not displayed as you click, it is possible that Attach Object to Mouse is disabled. To restore Click to Create mode:
     a. Click the Create *object* tab. The Create *object* dialog box opens.
     b. Select the Attach Object to Mouse check box.
   - If tactical graphics are not displayed after you create them, check to see whether the display of tactical graphics has been disabled. For details, see "45.7. Displaying Tactical Graphics" on page 998.

3. Optionally, you can set the altitude of an object as you create it. For details, see "14.4.1 Setting Altitude Dynamically" on page 387.
4. Optionally, you can set the heading of an object as you create it. For details, see "14.5. Specifying an Object's Heading Dynamically" on page 389.
5. Optionally, you can set the pitch of some objects as you create them. For details, see "14.6. Specifying an Object's Pitch Dynamically" on page 389.

   If you want to specify any of the object's properties in the Create *object* dialog box while using Click to Create, you can do so. However, be aware that as you move the cursor into the dialog box, the object graphic will move also. Once you specify the desired properties, click on the terrain to create an object with those properties. The properties will be applied to any additional objects you create until you change them or close the dialog box.

6. For simulation objects and waypoints, right-click or press **Esc** to exit create mode.

14.3.2 Specifying an Object's Properties Before You Create It (Click to Locate)

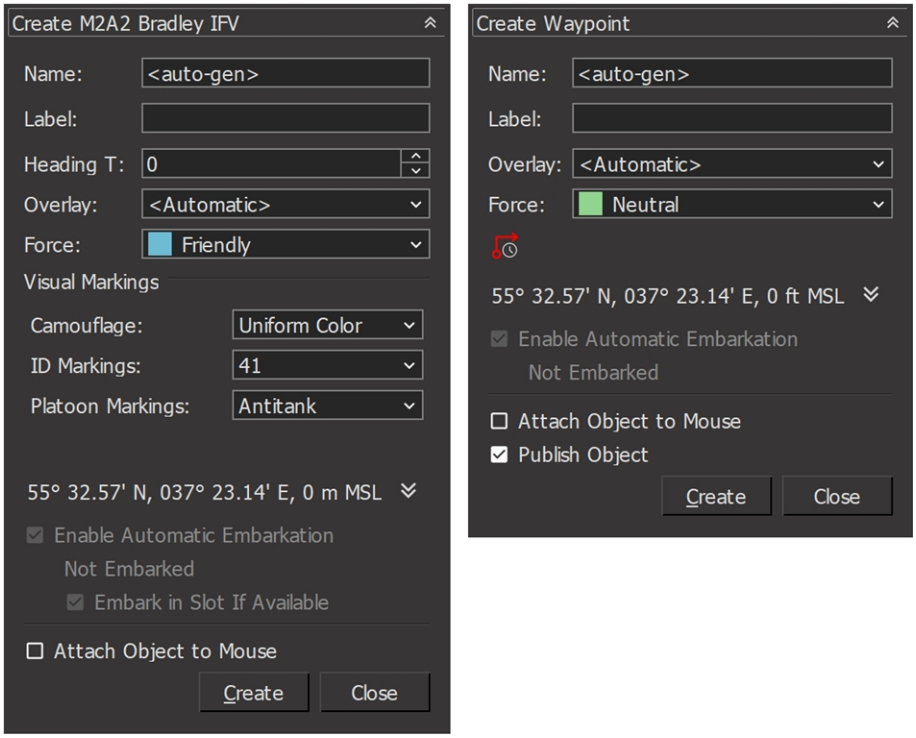
In the Click to Locate method, clicking on the terrain sets the coordinates of the object, but does not create it. You can specify all of the object's properties in the Create *object* dialog box before creating it. Table 22 on page 373 lists the properties that you can set for the different types of objects.

> VR-Forces remembers how you use the Create *object* dialog box on a per-object basis. For example, if you create a waypoint and open the Create Waypoint dialog box to specify its properties, the next time you create a waypoint, the Create Waypoint dialog box opens automatically. If the dialog box opens and you do not want it open for this object creation or for future creations of the current object type, click the Create *object* tab. The dialog box will close and VR-Forces will still be in object creation mode. The next time you create this object type, the dialog box will not open automatically.

➤ **To specify the properties of an object when you create it:**

1. Select the object that you want to create as described in "14.2. Selecting the Object to Create" on page 373. A Create *object* tab is added to the window below the palette (Figure 137 on page 381).
2. Click the Create *object* tab. A Create *object* dialog box that is appropriate for the object opens (Figure 140 below).

Figure 140. Create *object* dialog boxes



3. Clear the Attach Object to Mouse check box. For details about this option, see "14.7. Locking the Mouse to the Object Being Created" on page 390.
4. To specify the object's name, type a name in the Name box. For details about how simulation objects are named, see "13.2. How Simulation Objects Are Identified" on page 362.
5. Optionally, specify a label.
6. Optionally, if extended labels are available, specify them. For information about extended labels, see "21.3. Using Extended Labels" on page 477.
7. To specify the object's heading, enter a value, in degrees, in the Heading box. (Simulation objects only.)
8. For aircraft, you can enter the Current Speed and specify whether that speed is the ground speed, true airspeed, indicated airspeed, or Mach number.
9. To specify the overlay on which to place an object, select an overlay from the Overlay list.
10. Specify a force.
11. Optionally, if visual markings are available, specify them.
12. To specify a simulation object or point object's coordinates, click on the terrain or type coordinate values in the dialog box. To specify the altitude, follow the procedure in "14.4. Specifying an Object’s Altitude" on the next page.

    To specify the vertices of a route or area:

    a. Click on the terrain to place the first vertex, or type the coordinates in the dialog box.
    b. Click the Add button (+) to add a vertex.
    c. Click to locate the vertex or type the coordinates.
    d. Continue adding vertices until you are finished specifying the route or area. You can insert vertices anyplace in the object by selecting an existing vertex and clicking the Add button.

    To draw circles, ellipses, boxes, or text, see "Chapter 47. Creating and Editing Tactical Graphics and Control Objects" on page 1011.
13. For tactical graphics, specify whether to publish the object. For information about publishing tactical graphics, see "47.1.12 Publishing Tactical Graphics" on page 1019.
14. Click Create. The object is created. You can continue to create additional objects of this type with the same properties.
15. To close the dialog box, click Close.

14.3.3 Default Placement of New Entities

By default, ground, lifeform, rotary-wing, and fixed-wing entities are placed on the ground. Surface and subsurface entities are created at sea level.

Ground-based entities are placed at the highest possible terrain intersection at the location. If a terrain supports buildings with multiple levels, the entity is placed at the highest level. Once the entity begins moving, if the next calculated location resides in-between two levels of terrain, and if the entity is configured to respect terrain intersections when moving, and there is no terrain blocking its movement, the entity places itself at the closest elevation to the desired location.

14.3.4 Placing Entities Inside Buildings

One of the benefits of creating entities in a 3D observer mode is the ease of placing them in the precise location and orientation that you want them to be in, particularly inside buildings. However, placing entities inside small enclosed spaces can be hindered by the location of walls relative to the observer. You can work around this problem by adjusting the near clipping plane to remove the display of walls and floors until you see the location where you want to place entities. For details about changing the clipping plane, see "83.4.1 Setting the Clipping Planes" on page 1605.

> If you enable view constraints, it restricts your ability to move the observer into buildings. See "60.10. Enabling and Disabling View Constraints (3D Only)" on page 1200.

14.3.5 Placing Simulation Objects on Other Simulation Objects

If you want to place a simulation object on or in another simulation object, such as an aircraft on an aircraft carrier or a dismounted infantry entity in a truck, then you must embark the simulation object. Creating a simulation object on top of another simulation object will not provide the results you want. For example, if you create a fixed-wing entity and manipulate its location and altitude so that it looks like it is on the deck of an aircraft carrier, as soon as you start the simulation, the jet will start flying at that altitude because it is not attached to the carrier. It is simply created at a particular location. For complete details about embarking simulation objects, see "15.1. Embarking and Disembarking Simulation Objects" on page 406.

14.4. Specifying an Object's Altitude

You can specify an altitude for air platforms, sub-surface entities, the vertices of waypoints and routes, and props when you create them or edit their properties. (You can also set altitude for simulation objects using the Location set data request, Fly Altitude tasks, and the Move to Location task. You can set altitude dynamically or in the various dialog boxes for setting object properties or actions.)

14.4.1 Setting Altitude Dynamically

You can set altitude dynamically for simulation objects that support an altitude, for waypoints, for individual vertices of a route, and for props that have not been saved into a terrain. When you set altitude dynamically, it is set relative to the terrain.

➤ **To set altitude dynamically:**

- If you are creating an object, press and hold **Alt** and scroll the mouse wheel, or press and hold **Alt**+left mouse button and drag the mouse.
- If you are editing a simulation object or vertex:
  1. Double-click the object to make it editable.
  2. Press and hold **Alt** and scroll the mouse wheel, or press and hold **Alt**+left mouse button and drag the mouse.
  3. Click to set the new altitude.

ⓘ Optionally, you can press **Ctrl**+**Alt** and then click and drag to move the object by 5 meters at a time.

14.4.2 Setting Altitude in the Create Object or Edit Object Dialog Box

When you set altitude in a dialog box, you have the following options:

- Above Sea Level. The distance above sea level. If the terrain is higher than the altitude entered, the entity will be below the surface of the ground.
- Above Terrain. The distance above the terrain at this location.
- If you are creating or editing a route, you can adjust the altitude of all of the vertices at the same time and in the same way. For details, see "14.4.3 Specifying the Altitude for All of the Vertices in a Route" on the next page.

➤ **To set altitude in a dialog box:**

1. If you are creating or editing an object:
   a. Click the Create *object* (or Edit *object*) tab. A dialog box opens.
   b. Clear the Attach Object to Mouse check box.
2. Expand the location group box by clicking the arrow next to the coordinates and altitude.
3. Under Altitude, select one of the options.
4. Enter an altitude in the box.

14.4.3 Specifying the Altitude for All of the Vertices in a Route

When you create a route or edit its vertices, you may want to change the altitude of all of its vertices in a similar way. VR-Forces lets you:

- Adjust all vertices by the same amount.
- Set all vertices to the same height above sea level.
- Set all vertices to the same height above the terrain.

➤ **To change the altitude for all of the vertices in a route:**

1. If you are creating a route, open the Create Route dialog box.

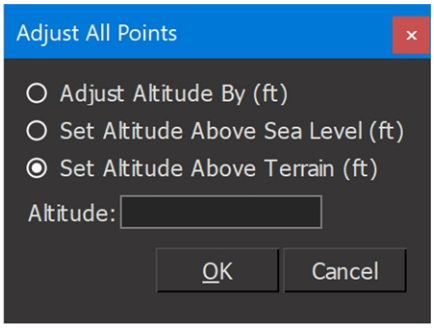
   If you are editing a route:

   a. Select the route.

   b. Choose **Objects > Edit**. The Edit Route dialog box opens.

2. Click the Adjust Points button (). The Adjust All Points dialog box opens (Figure 141).

Figure 141. Adjust All Points dialog box



3. Select the desired adjustment action:
   - Adjust Altitude By. Changes the altitude of each vertex by the amount specified in the Altitude box.
   - Set Altitude Above Sea Level. Sets the altitude of each vertex to the amount above sea level specified in the Altitude box. This could result in some vertices being below ground.
   - Set Altitude Above Terrain. Sets the altitude of each vertex the amount above the terrain specified in the Altitude box.
4. Enter the desired altitude adjustment value in the Altitude box.
5. Click OK.

14.5. Specifying an Object's Heading Dynamically

When you create a simulation object, cultural feature, or prop, you can set its heading in the Create *object* dialog box, or dynamically. Afterward, you can edit a simulation object or cultural feature's heading directly or use the Heading set data request. For more information, see "40.39. Heading" on page 882.

> In the 2D view, as you change the heading for a simulation object you can see the heading indicator change heading.

- To set an object's heading dynamically as part of the creation process, hold down the **Shift** key and the left mouse button, then drag the mouse.
- To edit the heading of an object that has already been created, double-click it to put it into edit mode. Then hold down the **Shift** key and the left mouse button, then drag the mouse.

> Optionally, you can press **Ctrl**+**Shift** and then click and drag to adjust the heading by 15 degrees at a time.

14.6.  Specifying an Object's Pitch Dynamically

In entity-level scenarios, some simulation objects, cultural features, and props allow you to set the pitch in the Create *object* dialog box or dynamically. Afterward, you can edit the object's pitch directly or use the Pitch set data request. For details, see "40.58. Pitch" on page 892.

> There is no indication of the pitch in the 2D view, therefore you may want to switch to see the model's pitch and adjust it as needed.

- To set an object's pitch dynamically as part of the creation process, press and hold **Shift**+**Alt** and the left mouse button, then drag the mouse.
- To edit the pitch of an object that has already been created, double-click it to put it into edit mode. Then press and hold **Shift**+**Alt** and the left mouse button, then drag the mouse.

> Optionally, you can press **Ctrl**+**Shift**+**Alt** and then click and drag to adjust the pitch by 15 degrees at a time.

14.7. Locking the Mouse to the Object Being Created

By default, when you create simulation objects or waypoints, you can create multiple objects by continuously left-clicking without exiting create mode. This is a very convenient feature if you want to create objects using default naming conventions and do not need to specify specific location coordinates or overlay assignments during the creation process. (You can always edit an object's properties after you create it.)

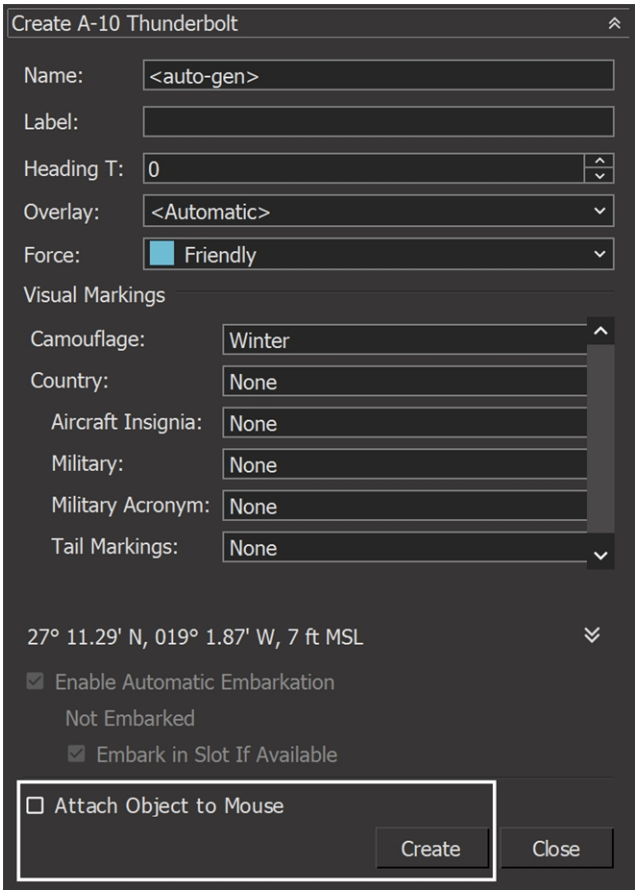
However, if you want to use the Create *object* dialog box to specify location or other properties for each object as you create it, this feature can be inconvenient because as you move the mouse to the Create *object* dialog box, the placement location changes with the mouse movement. Disabling this feature can give you greater control over the creation process, because clicking to set the location of the object or a vertex does not create it – it sets the coordinates, but lets you make further changes in the Create *object* dialog box before you create the object.

If you disable an object's attachment to the mouse when you are creating objects, you create them by clicking Create in the dialog box rather than clicking on the terrain.

> If you have cleared the Attach Object to Mouse check box in the Create *object* dialog box, you must still specify the minimum number of points for the object before the Create button is enabled. For simulation objects this is only the location of the object, but for tactical graphics such as lines, circles, and areas, you may need to specify two, three, or more points. Since the object is not attached to the mouse, you will not see dynamic resizing of the object as you specify the points.

➤ To enable or disable the mouse's attachment to the creation location, in the Create *object* or Edit *object* dialog box, select or clear the Attach Object to Mouse check box (Figure 142). When you clear the check box, a Create button is added to the dialog box. The setting is carried over to future creations of that object.

Figure 142. Create *object* dialog box



14.8. Simulation Object Resources

When you create a simulation object, it has a full complement of resources, as defined in the object parameter database. You do not have to assign resources to a simulation object when you create it. However, if you want a simulation object to start a simulation with less than a full set of resources, you can use the Resources set data request to set its resources at some amount less than 100%. For details about the Resources set data request, see "40.67. Resources" on page 896.

14.9. Moving Objects

You can move an object by dragging it, changing its location in the Edit dialog box or, for simulation objects only, sending a Location set data request.

For information about setting the location of a simulation object, see "40.49. Location" on page 887.

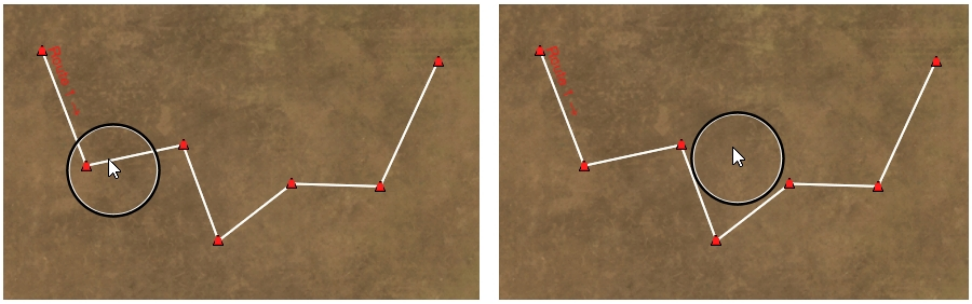
You can only move objects simulated by VR-Forces.

14.9.1 Dragging an Object to a New Location

When you move objects, they are centered on the mouse location:

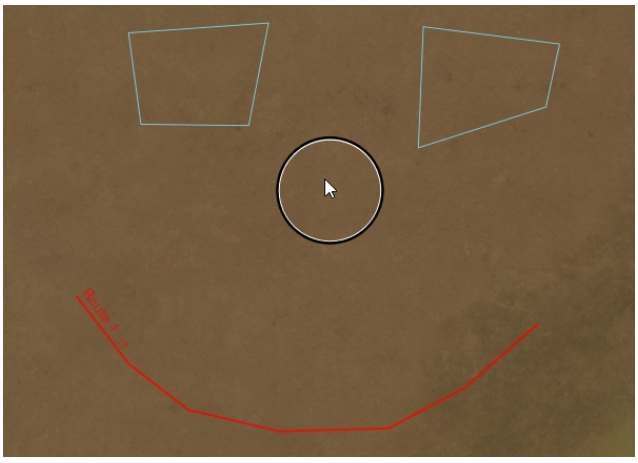
- If you select an object by double-clicking, it attaches to the mouse at the location where you clicked it. (For simulation objects and point objects, the objects are centered on the mouse location, but for lines or areal objects, the attachment is at the point where you clicked (left image in Figure 143).)
- If you select an individual object and move it by choosing the Move command:
  - Point objects are centered on the mouse, just as when you double-click them.
  - Linear and areal objects attach to the cursor at the center of mass (right image in Figure 143).

Figure 143. Mouse attachment when moving individual lines



- If you select multiple objects and move them by choosing the Move command:
  - Point objects attach to the cursor at their center of mass.
  - Multiple objects have their centers of mass averaged and the center point attaches to the cursor (Figure 144).

Figure 144. Moving multiple objects



If you click the terrain and then **Shift**+click or **Ctrl**+click an object, the terrain is selected and the Move command will not be available.

➤ **To drag an object from one point on the terrain to another:**

1. Double-click the object, or select the object and choose **Objects > Move**.
2. Move the mouse to change the location of the object.
3. Left-click the mouse.

- You can drag multiple objects and they can be of different types.
- To drag multiple objects, you must select the objects and then choose **Objects > Move**.
- There may be indicators showing that the object is moving from its original location to the new one. This tells you that the move was successful while VR-Forces repositions the object.

Canceling a Drag-and-Drop Move

➤ To cancel a move made by dragging an object, press **Esc** before you click the left mouse button.

14.10. Editing Simulation Objects

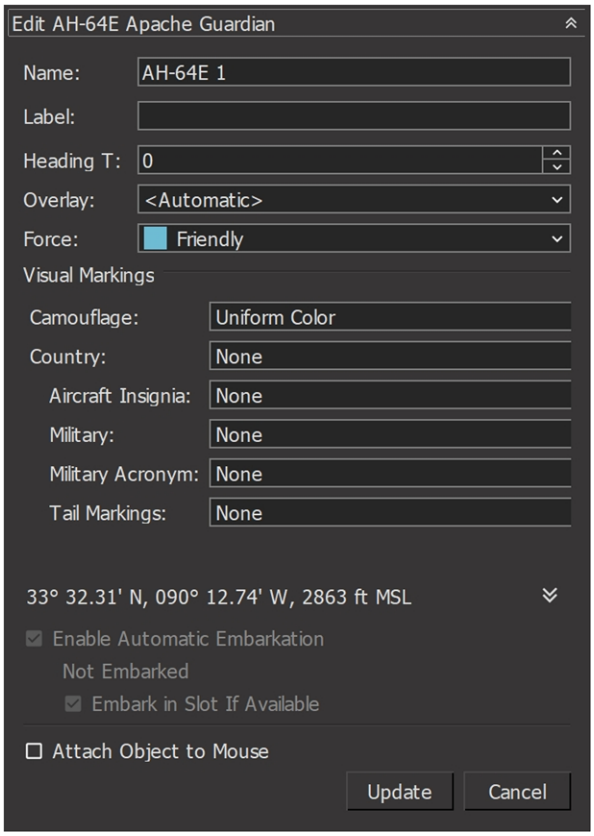
You can edit these simulation object properties:

- Name
- Label (and extended labels if available)
- Force
- Visual Markings (if available)
- Location
- Heading
- Altitude
- Overlay

➤ **To edit a simulation object:**

1. Select the simulation object.
2. Choose **Objects > Edit**. The Edit Object dialog box opens (Figure 145).

Figure 145. Edit Object dialog box



3. Change the properties as desired.
4. Click OK.

You can also change the location, altitude, and heading dynamically, through a variety of movement tasks or with set data requests. For details, see:

- "14.4.1 Setting Altitude Dynamically" on page 387.
- "14.5. Specifying an Object's Heading Dynamically" on page 389.
- "14.9. Moving Objects" on page 391.
- "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583.
- "Chapter 35. Tasks for Aggregate-Level Scenarios" on page 711.
- "Chapter 40. Setting the State of Entity-Level Simulation Objects" on page 863.

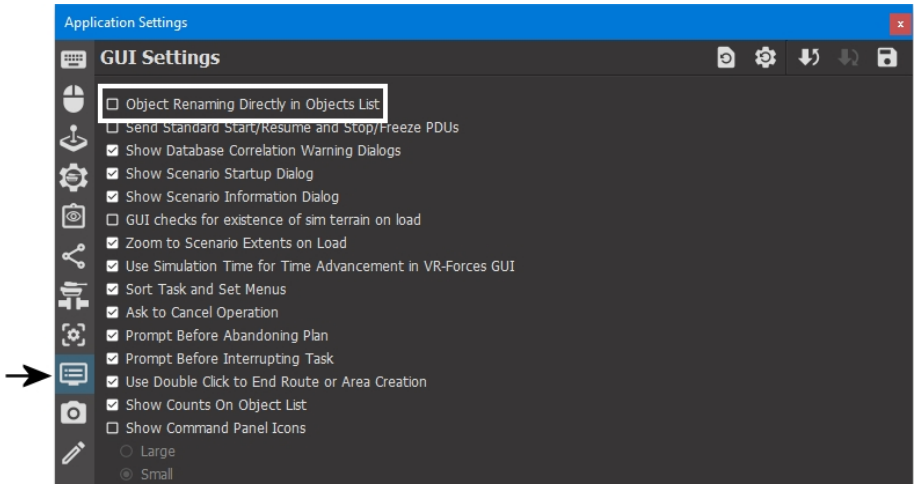
14.10.1 Editing an Object's Name in the Object List

You can edit an object's name in an Edit dialog box or directly in an object list. The ability to edit an object's name in an object list is disabled by default. This prevents you from inadvertently going into edit mode when you want to select an object.

➤ **To enable and disable editing an object's name in an object list:**

1. Choose Settings Application Settings. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 146).

Figure 146. GUI Settings page



3. To allow editing of names in an object list, select the Object Renaming Directly in Objects List check box. To prevent editing of names, clear the check box.
4. Click Close

➤ **To edit an object's name in an object list:**

1. Click the name of the object, then click again. (This is two single clicks, not a double-click. The two clicks should be about one second apart.) You can also press F2. The name becomes editable.
2. Edit the name.
3. Click someplace else in the window.

14.11. Copying and Pasting Objects

You can copy objects and paste the copies elsewhere in a scenario or even in another scenario. The Paste Special option allows you to specify which entity characteristics you paste.

VR-Forces copies the following data about an object:

- Environmental objects:
  - Original name
  - Label
  - Force
  - Color

- Entities:
  - Original name
  - Label
  - Force
  - Heading
  - Appearance (lights, landing gear, and so on)
  - Rules of engagement
  - Plan
  - State property values
- Units copy the same information as entities as well as the list of subordinates.

VR-Forces does not copy:

- The current task.
- State information, except as noted previously.
- Load balancing status. All objects are pasted into the currently selected sim engine.
- Embarkation structures. If, for example, you copy an aircraft carrier that has planes embarked, only the carrier is copied, not the embarked entities.

> ⚠ The copy/paste process is resource intensive. It is primarily for use during scenario development. Use during a simulation could affect performance.

14.11.1 Copying Objects

➤ **To copy objects:**

1. Select the objects you want to copy.
2. Choose **Objects > Copy**.

14.11.2 Pasting Objects

A simple paste pastes all of the attributes of an object that were copied. It pastes objects as follows:

- Tactical graphics are placed on the currently selected overlay. If you copied objects that were on different overlays, that differentiation is not preserved.
- Copied objects maintain the state at the time that they were copied. Therefore, if you paste objects some time after they were initially copied, the copies do not reflect any changes that may have taken place to the original simulation objects. For example, if a simulation object was copied and then deleted, the copied version of the entity is still pasted.
- Embarkation structures are not copied, only the parent entity.

> ⚠ Pasting plans may have unexpected consequences because their object references do not change when you copy them. For example, suppose the plan for Entity A refers to Waypoint 1. You copy Entity A and Waypoint 1 and paste them as Entity B and Waypoint 2. If you copy the plan, Entity B's plan will still refer to Waypoint 1. If you expected it to refer to Waypoint 2, you will not see the behavior you expected. You can choose not to paste plans. For details, see "14.11.3 Pasting Specific Entity Characteristics" below.

➤ **To paste objects:**

1. Choose **Objects > Paste**. The objects in the paste buffer display on screen. They float as you move the cursor to show you where they will be placed when you paste them.
2. Click on the terrain where you want the objects to be pasted. They are pasted in the relative positions in which they were copied. If an object has an altitude, it is pasted using the same altitude above the terrain as the original object.
3. Optionally, continue pasting additional copies if desired.
4. Right-click to exit paste mode.

14.11.3 Pasting Specific Entity Characteristics

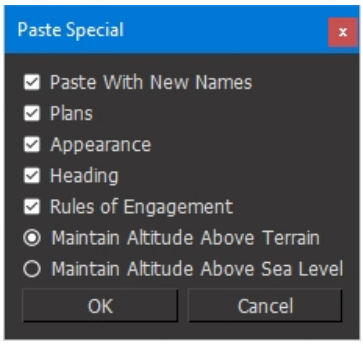
When you paste simulation objects, you can specify which of these entity characteristics you want to paste with the entity:

- Plans
- Appearance
- Heading
- Rules of engagement
- Entity name
- Altitude. You can specify that the previous altitude be applied as above terrain or as above sea level.

➤ **To paste particular entity characteristics:**

1. Choose **Objects > Paste Special**. The Paste Special dialog box opens.

Figure 147. Paste Special dialog box

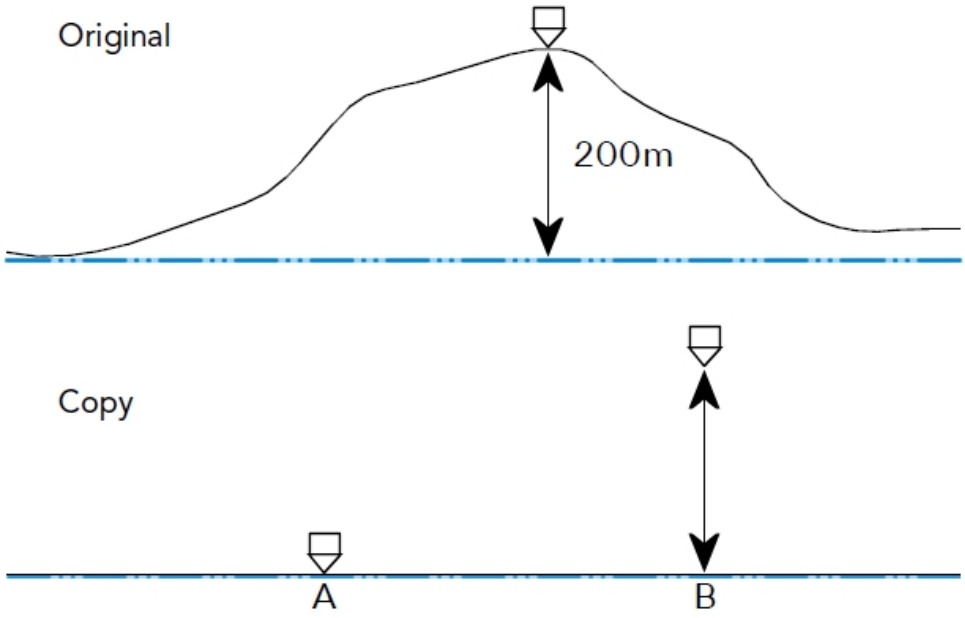


2. By default, all check boxes are selected. Clear the check boxes for the options you do not want to paste.
3. Choose the altitude option that you want to use.
4. Click OK. The objects in the paste buffer display on screen. They float as you move the cursor to show you where they will be placed when you paste them.
5. Click on the terrain where you want the objects to be pasted. They are pasted in the relative positions in which they were copied.
6. Optionally, continue pasting additional copies if desired.
7. Right-click to exit paste mode.

Pasting Altitude

When you paste objects, if the pasted object can have an altitude value, such as a fixed-wing entity, a waypoint, or the vertices of a route, the default behavior is to replicate the original altitude above the terrain. If you use the Paste Special operation, you can choose to apply the original altitude as being above sea level. Figure 148 illustrates these alternatives. A waypoint is placed on the terrain. It is 0 meters above the terrain. However, the terrain itself is 200m above sea level. If the waypoint is copied and pasted at point A and the altitude above the terrain is maintained, the point is placed at 0m above the terrain. If the point is pasted at point B and altitude above sea level is maintained, it is placed 200m above sea level.

Figure 148. Pasting altitude



14.12. Deleting Simulation Objects

➤ **To delete a simulation object:**

1. Select the object you want to delete.
2. Choose **Objects > Delete** or press the **Delete** key.

- If you delete a unit, all of its subordinates are also deleted.
- If you delete a simulation object that has an intervisibility object pinned to it, the intervisibility object is also deleted. For details, see "55.9. Deleting Intervisibility Objects" on page 1130.

14.13. Reinitializing Simulation Objects

You can reinitialize an individual simulation object, which resets the object using the parameters based on the SMS. It clears any process state information. Use this if there were changes to the entity's parameters after you created the scenario, for example, a correction to the system definition (*.sysdef*).

This is different than reinitializing the entire scenario. To learn more about that, see "7.7. Reinitializing a Scenario" on page 264.

➤ **To reinitialize a simulation object:**

1. Select the object you want to reinitialize.
2. Choose **Objects > Reinitialize**.

14.14. Simulation Object Groups

Simulation object groups (SOG) are a convenient mechanism for creating reusable configurations of simulation objects and tactical graphics. You create them in the Simulation Object Editor and they are listed on the Create Object Palette with other simulation objects. VR-Forces places them in the relative positions that you specified when you created the SOG. VR-Forces has example SOGs, including a carrier group, a Patriot missile battery, a barracks with several soldiers, and others. These examples demonstrate SOGs with tactical graphics, cultural features, and embarkation.

> The Barracks With Guards SOG has some tactical graphics, but they are hidden by default. To see them, click the overlay icon at the bottom of the window, or select the check box on the Objects List Panel, Overlays View.

The Create Object Palette has a tag for SOGs (Object Group), but like other simulation objects, they can have multiple tags. SOGs can be distinguished from other simulation objects in the lists because the object name includes the number of objects in the group.

For information about creating SOGs, see "69.6. Creating a Simulation Object Group" on page 1362.

14.15. Creating Random Versions of Similar Entity Types

VR-Forces has many different models for generic types of entities, such as civilian vehicles or commercial aircraft. When you are creating a scenario with these sorts of entities, you can choose the specific entity types you want to use. However, if you want to quickly populate a scenario with a variety of similar objects, it is simpler to have VR-Forces randomly create instances of the generic entity. This is done using the random simulation object.

A random simulation object is configured in the Simulation Object Editor with a list of specific simulation object types that it can draw from. When you create a scenario, you choose the random entity type on the Create Object Palette and when you click on the terrain to place the object, VR-Forces picks a specific simulation object from the list.

VR-Forces includes a random simulation object for civilian vehicles.

You can add your own random simulation objects in the Simulation Object Editor. See "70.7. Creating Randomized Simulation Objects" on page 1396.

14.16. Creating Cultural Features

Cultural features are objects that you can add to a scenario to simulate human artifacts such as buildings, bridges, and so on. They have the Cultural Feature tag on the Create Object Palette. You add them to a scenario the same way you would add an entity or tactical graphic. Cultural features are listed in the Objects List View of the Objects List Panel. You can apply some set data requests to them, such as Location and Heading.

> ⚠ Some of the choices on the cultural feature lists are the same as those available on the Props Palette. However, cultural features are not the same things as props. Props are terrain objects and can be saved as part of a terrain. Cultural features are not part of the terrain; they are a type of simulation object and are saved in the scenario objects file.

14.16.1 Embedding Cultural Features in the Terrain

If you have a cultural feature, such as a house, that has object geometry, VR-Forces can treat that object as part of the terrain when you place it in a scenario. This means that entities can walk on it or in it (if it has interior geometry).

If you place the object outside of a navigation area, entities interact with it similarly to how they would interact with a prop. If you place the object in a navigation area, VR-Forces dynamically regenerates navigation data for the sectors in which the cultural feature is located. Entities can then use the navigation data to walk around or in the cultural feature more intelligently than if it was outside of a navigation area.

> ⓘ You do not need a license for generating navigation areas to dynamically regenerate navigation data. However, you must have a terrain on which navigation areas have been created, such as the terrains provided with VR-Forces. For information about navigation areas and navigation data, see "66.3. Generation of Navigation Data" on page 1281.

This feature has the following requirements and constraints:

- The cultural feature must be configured in the Simulation Object Editor to use object geometry in the simulation engine and to be embedded in the terrain. For details, see "70.1.7 Configuring Cultural Features To Be Embedded in the Terrain" on page 1381.
- If you place the cultural feature in a navigation area, the area must be sectorized. If it is not sectorized, the navigation data will not be generated.
- Navigation data is generated dynamically. It is not saved with the terrain or with the scenario. When you reload the scenario, the navigation data is generated after the scenario loads. If you have many such cultural features in a scenario, this can take some time. During that time, you might not be able to use VR-Forces or other applications on your computer.
- Creating a navigation area and generating navigation data for a portion of a terrain in which you have placed a cultural feature does not generate navigation data for the cultural feature. The navigation data is generated based on the content of the terrain and the cultural feature is a simulation object, not part of the terrain. If you want the geometry of a cultural feature to be part of the terrain, add it as a prop.

➤ To embed a cultural feature in the terrain, create it in a scenario.

14.17. Creating Props

Props are terrain objects that you can select independently of the terrain. "Extracting Props from a Terrain", in the *Adding Content to MAK ONE Applications Reference Manual*, explains how to create them by extracting them from a terrain or a feature layer. You can also create them from the Props Palette, however, similarly to how you create simulation objects. This is a valuable feature if your terrains do not have any features that you can extract as props. Do note these important details:

- Props are not saved in a scenario. If you want to save a prop, you must save the terrain. You can save the terrain to a different name or overwrite the existing terrain. The next time you open the terrain, the prop will be part of it.
- When you create a prop, the sim engine does not know anything about it because it is a terrain object. To have the sim engine take the prop into account, you must save the terrain (as mentioned in the previous paragraph) and restart the scenario.
- Props are created on the GUI terrain. If the GUI and sim engine are using different terrains, the sim engine terrain will not include the props.
- You can export props to a shapefile, allowing you to add them to a terrain using an earth file. For details, see "Exporting Props and SpeedTrees to a Shapefile" in the *Adding Content to MAK ONE Applications Reference Manual*.
- You can also create cultural features that look exactly like the props that you can create. However, cultural features are not the same things as props. Cultural features are treated as simulation objects. They are listed with simulation objects and tactical graphics and are saved as part of the scenario.

> The list of props that you can create from the Props Palette is based on the model definitions defined on the Model Definitions tab of the Visual Model Editors dialog box. Therefore, you can add new props to the Props Palette by adding new model definitions. For details, see "Adding Object Models as Props" in the *Adding Content to MAK ONE Applications Reference Manual*.

14.18. Adding an Object to the Favorites List

As mentioned in "14.2. Selecting the Object to Create" on page 373, the Create menu does not include all of the objects that you can create. It just lists the objects that you have designated as "Favorites". When you specify that a simulation object or tactical graphic be a favorite, it is automatically added to the Create menu. An option is added to the menu for the type associated with the favorite. As you add additional favorites, they are listed under all of the tags that apply to them.

In the Create Object Palette, objects that are favorites have a small star next to the icon. You can also filter the object list using the Favorites tag (★) to see which objects you have marked as favorites.

The favorites list is specific to a simulation model set. It is saved to *./data/simulationModelSets/<model_set>/gui/favorites.mst*. You can copy the file to other instances of VR-Forces or future releases and your favorites list will be enabled.

➤ **To add or remove an object from the Favorites list:**

1. Open the Create Object Palette.
2. Right-click the object.
3. On the pop-up menu, select **Add to Favorites**.

14.18.1 Removing an Object from the Favorites List

When you remove an object from the Favorites list, it is also removed from the Create menu.

➤ **To remove an object from the Favorites list:**

1. Open the palette that has the object you want to remove from the Favorites list.
2. Right-click the object.
3. On the pop-up menu, select **Remove from Favorites**.



15. Embarked and Embedded Entities

This chapter explains how to embark entities on other entities. It also explains how to manage embedded entities, which is another way to embark entities.

15.1. Embarking and Disembarking Simulation Objects 406
15.1.1 Embarking and Disembarking Simulation Objects Instantly 407
15.1.2 Disembarking Simulation Objects Using the Disembark Command 410
15.2. Embedded Entities 411
15.2.1 Configuring an Entity to Support Embedded Entities 411
15.2.2 Assigning Tasks and Plans to Embedded Entities 413
15.2.3 Restoring an Entity that has Embedded Entities 414

15.1. Embarking and Disembarking Simulation Objects

Embarkation places simulation objects into or onto other simulation objects, for example, dismounted infantry in a truck, trucks on an LCAC, or aircraft on an aircraft carrier. While embarked, the simulation objects travel with the simulation object on which they are embarked. Embarked simulation objects are not visible in the 2D view. They are visible in the 3D view if they are embarked on the outside of the parent, such as on the deck of an aircraft carrier, or inside a parent that has interior geometry, such as some helicopters.

Embarked entities can shoot while they are embarked if they are embarked on an entity that has object geometry enabled.

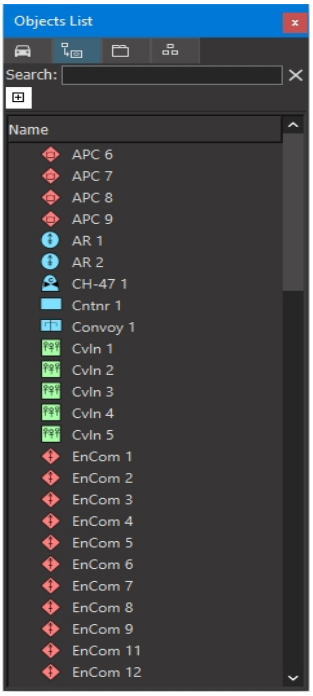
> ⚠ If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later. VR-Forces includes configurations in the Simulation Connections Configuration dialog box that start VR-Forces using the correct FED file and FOM Mapper for HLA Evolved using RPR FOM 2.

You can attach control objects to simulation objects so that embarked simulation objects can use them to move about on the parent simulation object. For example, an aircraft could taxi towards a waypoint on the flight deck of an aircraft carrier.

> ⓘ Attached objects are shown in the Embarkation View on the Objects List Panel.

The embarkation status of simulation objects is displayed in the Embarkation View of the Objects List Panel (Figure 149).

Figure 149. Embarkation View



You can embark and disembark simulation objects instantly (described in this section), or you can have them simulate embarkation and disembarkation using the Embark and Disembark tasks, described in "Chapter 34. Embarkation, Wait, Radio, and Other Tasks" on page 679.

When you embark a simulation object using the Embark task, the simulation object can embark only on simulation objects that are configured to accept that type of simulation object. For example, LCACs are configured to accept several types of ground vehicles. You cannot give a DI an Embark task to embark on an LCAC. However, when you use the immediate embark method to embark a simulation object, you can use preconfigured slots (spaces) for simulation objects or specify an override. When you specify an override you can embark a simulation object on any other simulation object at the coordinates you enter. However, it is your responsibility to know what the proper coordinates are.

Embarked entities controlled by joystick cannot disembark.

15.1.1 Embarking and Disembarking Simulation Objects Instantly

You can embark and disembark simulation objects instantly by using:

- The Embark On command.
- Set data requests.
- The Embarkation View.
- Automatic embarkation.

Using set data requests lets you include immediate embarkation in plans. This may be particularly useful in a global plan, in which you may want to quickly create and embark simulation objects without paying attention to the realism of the action. For details about using set data requests for embarkation, see "40.28. Embarked" on page 877 and "40.27. Disembarked" on page 877.

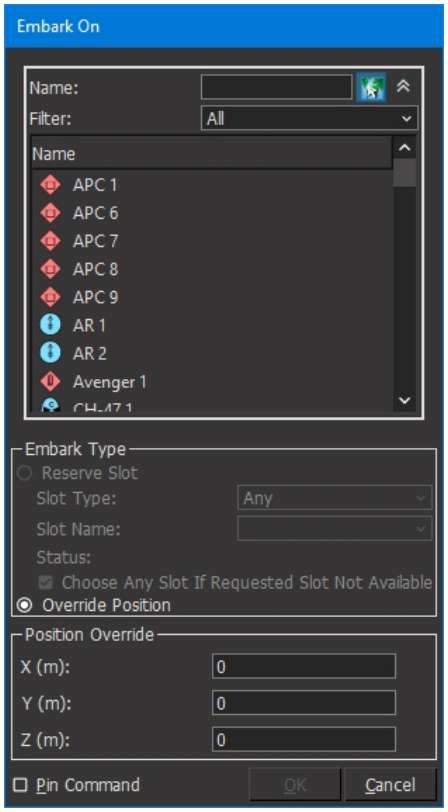
Embarking Simulation Objects Instantly Using the Embark On Command

The Embark On command lets you embark simulation objects using preconfigured embarkation spaces or using specific coordinates. If the embarkation spaces on the parent simulation object are full, VR-Forces sends a message to the sim engine console and the simulation object is not embarked. To embark simulation objects instantly from a plan, use the Embarked set data request.

➤ **To embark a simulation object instantly using the Embark On command:**

1. Select the simulation object that you want to embark.
2. Choose **Objects > Embarkation > Embark On**. The Embark On dialog box opens.

Figure 150. Embark On dialog box



3. Optionally, filter the list of potential parent simulation objects.
4. Select the parent simulation object.
5. To embark in a configured space, select the Reserve Slot option. To specify an embarkation point, select Override Position.
6. If you chose the Override Position option, specify the coordinates on the parent, in body coordinates, for the embarkation point.
7. Click OK.

Embarking Simulation Objects in the Embarkation View

➤ **To embark an object in the Embarkation View:**

1. On the Objects List Panel, select the Embarkation View.
2. Drag the simulation object you want to embark onto the parent simulation object. The Embarked dialog box opens.
3. To specify an embarkation location, specify X, Y, and Z values. If you do not specify a location, the simulation object embarks at the center of the parent simulation object.
4. Click OK.

Embarking Simulation Objects Using Automatic Embarkation

Automatic embarkation lets you embark simulation objects as you create them. It has these conditions:

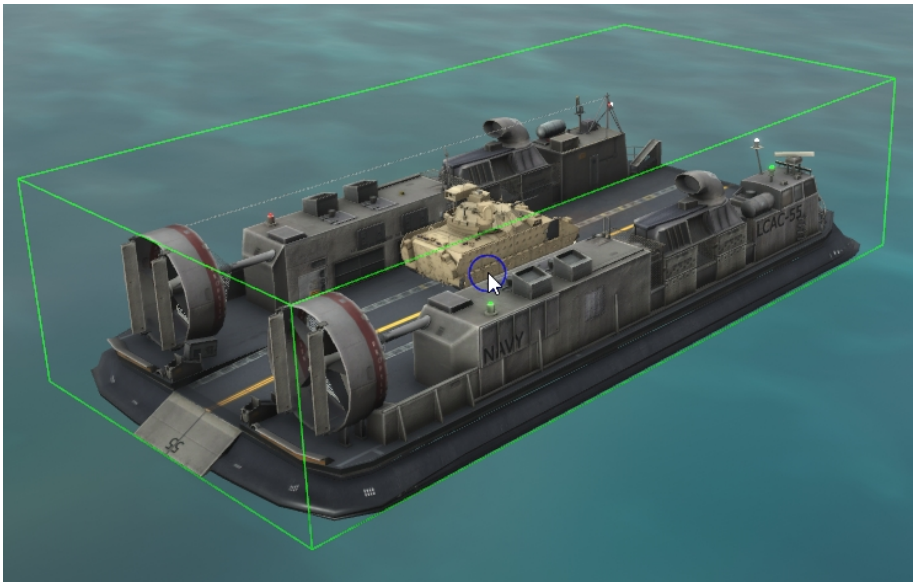
- It is only available in 3D observer modes.
- The simulation object you want to embark must be capable of embarking (as configured in the OPD).
- The simulation object you want to embark on must allow embarkation.
- There is no option to override embarkation slots by specifying a particular location.
- If slots are configured on the parent simulation object and you select the option to use slots, embarked simulation objects are placed in slots until the slots run out. Then they are placed on the parent simulation object where you click.

You can attach tactical graphics to simulation objects using this method. For more information, see "47.8.1 Attaching a Tactical Graphic Automatically" on page 1029.

➤ **To embark a simulation object using automatic embarkation:**

1. Create the entity on which you want to embark another entity (if it is not already created.)
2. In the Create Object Palette, select the entity that you want to embark.
3. Optionally, configure auto-embarkation:
    a. If the Create Entity dialog box is not open, click its tab to open it.
    b. To enable or disable automatic embarkation, select or clear the Enable Automatic Embarkation check box.
    c. To embark the entity in a slot if one is available, select the Embark in Slot if Available check box.
4. Move the cursor over the entity you want the new entity to embark on. If it supports embarkation and automatic embarkation is enabled, a green box is displayed (Figure 151). (If the entity does not support automatic embarkation, no green box is displayed.) If the create dialog box is open, the Not Embarked label changes to Embark On *entity_name*.

Figure 151. Auto embarkation



5. Click to place the entity.

15.1.2 Disembarking Simulation Objects Using the Disembark Command

Like the Embark On command, the Disembark command is typically used as part of scenario development. To simulate disembarkation, you would use the Disembark task. To provide instant disembarkation in a plan, you would use the Disembarked set data request.

Embarked entities controlled by joystick cannot disembark.

➤ **To disembark a simulation object instantly:**

1. Select the simulation object that you want to disembark.
2. Choose **Objects > Embarkation > Disembark**. If the simulation object was embarked using the default embarkation point, it is disembarked next to the former parent simulation object. If the simulation object was embarked using a position override, it is disembarked in the same location as the former parent.

➤ **To disembark a simulation object in the Embarkation View:**

1. Open the Embarkation View.
2. Drag the simulation object you want to disembark from the parent simulation object to the force level.

Disembarking All Simulation Objects

When you disembark all simulation objects from a parent, the placement of the simulation objects depends on how they were embarked. If the simulation objects were embarked with the Embark On command or the Embark task, they are disembarked to separate locations near the parent. If they were embarked with the Embarked set data request,

they are all disembarked in the same location in the center of the parent simulation object. Since this behavior is usually not desirable, you should understand how you embarked the simulation objects before you use this command.

➤ **To disembark all simulation objects from a parent entity:**

1. Select the parent simulation object.
2. Choose **Objects > Embarkation > Disembark All**.

15.2. Embedded Entities

Embedded entities are a convenient way to manage entities that are typically embarked on other entities.

Ships, planes, and vehicles often carry other platforms or personnel. You can simulate this relationship by embarking entities on other entities. However, this involves creating the embarking entities, giving them plans, embarking them, disembarking them, and otherwise managing the relationship. This can be inconvenient if you need to simulate large numbers of embarked entities, such as the airplanes and helicopters on an aircraft carrier or multiple personnel on a personnel carrier.

The embedded entity feature lets you configure entities with a set of embedded entities. You can then deploy the embedded entities and give them tasks or plans. Once deployed, there is no difference between an embedded entity and any other entity, except that a parent entity can recover a deployed entity. You do not need to manage the initial process of creating the relationship between the entities via embarkation and disembarkation.

For details about deploying and recovering embedded entities, see "34.8. Embedded Entity Tasks" on page 703.

15.2.1 Configuring an Entity to Support Embedded Entities

Embedded entities are configured in systems, which you then add to an entity in the Simulation Object Editor. An entity can have only one embedded entity system. Therefore, if you need different configurations of embedded entities for different parent entities, you must create a system for each configuration.

Within a configuration, you must specify each embedded entity. In other words, you cannot specify a system as containing, for example, 5 helicopters and 40 F/A-18 jets. In this case the system would have to individually specify each helicopter and each jet.

A system definition file for embedded entities specifies the entities and the metadata needed to edit properties in the Simulation Object Editor. You must give each entity a unique name.

An entity is formatted as follows (from *./data/simulationModelSets/EntityLevel /vrfSim/systems/other/ddg-embedded-support-craft.sysdef*):

```
(embedded-entities
  (ee-1
    (entity-name  "SH-60 Seahawk Alpha")
    (entity-type  1 (1 2 225 22 3 1 0))
    (deployment-time 0.0)
    (recovery-distance 35.0)
    (relative-position $rel-pos-lamps-a)
```

```
      (relative-heading $rel-heading-lamps-a)
    )
    ...
    (ee-2
      (entity-name  "RHIB Alpha")
      (entity-type  1 (1 3 225 63 8 0 0))
      (deployment-time 15)
      (recovery-distance 50.0)
      (relative-position $rel-pos-rhib-a)
      (relative-heading $rel-heading-rhib-a)
    )
    ...
  )
```

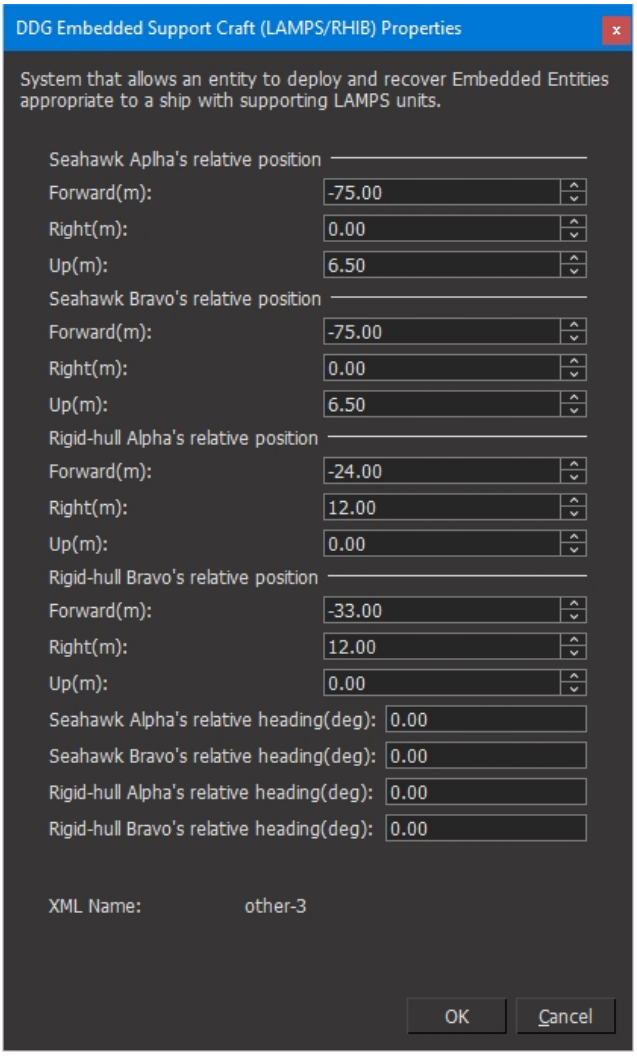
The **relative-position** and **relative-heading** parameters are variables that link to the metadata:

```
  (parameter-data-list
    (vector-parameter-data
      (parameter-name  "rel-pos-lamps-a")
      (variable-type  "DtRwVector")
      (display-label  "Seahawk Alpha's relative position")
      (display-units  "meters")
      (source-units  "meters")
      (default-value  -75.000000 0.000000 -6.500000)
      (relative-to  "")
    )
    (vector-parameter-data
      (parameter-name  "rel-pos-rhib-a")
      (variable-type  "DtRwVector")
      (display-label  "Rigid-hull Alpha's relative position")
      (display-units  "meters")
      (source-units  "meters")
      (default-value  -75.000000 10.000000 -6.500000)
      (relative-to  "")
    )
    ...
    (real-parameter-data
      (parameter-name  "rel-heading-lamps-a")
      (variable-type  "DtRwReal")
      (display-label  "Seahawk Alpha's relative heading")
      (display-units  "degrees")
      (source-units  "degrees")
      (default-value  0.000000)
    )
    (real-parameter-data
      (parameter-name  "rel-heading-rhib-a")
      variable-type  "DtRwReal")
      (display-label  "Rigid-hull Alpha's relative heading")
      (display-units  "degrees")
```

```
    (source-units  "degrees")
    (default-value  90.000000)
  )
  ...
)
```

Figure 152 illustrates the Additional Systems properties dialog box created for the system shown in the previous paragraphs.

Figure 152. Embedded Entity Properties dialog box



15.2.2 Assigning Tasks and Plans to Embedded Entities

Once an embedded entity is deployed, it is just like any other entity in the scenario. You can give it independent tasks or write a plan for it. Follow the standard procedures for these activities.

15.2.3 Restoring an Entity that has Embedded Entities

If you deploy an embedded entity and it is destroyed, it is removed from the scenario and is not available to be deployed again. However, if you restore a parent entity and it had deployed entities that were subsequently destroyed, it is restored to the full complement of embedded entities. Entities that are deployed at the time the parent is restored are not affected. They are not duplicated in the parent's inventory and their relationship is not affected.



16. Generating Traffic (Pattern of Life)

This chapter explains how to generate pattern of life simulation object traffic and how to create and task pedestrian crowds.

16.1. Generating Traffic 416
16.2. Spawn and Sink Traffic Generation 416
16.2.1 Creating Spawn Points 417
16.2.2 Creating Sink Points 418
16.2.3 Spawn Patterns 419
16.2.4 Spawn Pattern Management 422
16.3. Clutter Areas 428
16.3.1 Aircraft Clutter Area 428
16.3.2 Ship Clutter Area 429

16.1. Generating Traffic

You can make your scenario more realistic by including ground vehicles, pedestrians, surface watercraft, and aircraft that appear to be going about their own business. VR-Forces offers:

- A general-purpose spawn-and-sink traffic feature that can generate both vehicular traffic (including aircraft, watercraft, and ground vehicles) and pedestrian traffic (including humans and animals).
- One method for generating air traffic.
- One method for generating background watercraft.

Most of this chapter explains spawn-and-sink traffic generation. For a discussion of choosing a method of air traffic generation and instructions on using the air-traffic-only methods, see "16.3.1 Aircraft Clutter Area" on page 428. For information on creating a watercraft pattern of life based on historical data, see "16.3.2 Ship Clutter Area" on page 429.

16.2. Spawn and Sink Traffic Generation

The VR-Forces spawn and sink traffic feature (one of the pattern of life features) allows you to automatically generate background traffic for your scenario.

Automatically generated (spawned) simulation objects are created at spawn points. The default behavior for a spawned simulation object is to choose a destination, called a sink point, and move towards that point. If there are multiple sink points that it can move to, it randomly selects one. When the simulation object arrives at the sink point, the it is removed from the scenario. This creates the effect of purposeful activity in an area of the terrain without the need to create and task simulation objects individually.

As an alternative to the default behavior, you can write a plan that is assigned to each simulation object created at a particular type of spawn point. This plan can include any tasks and set data requests that are appropriate for the object type.

The behavior of a spawned simulation object, as well as other details such as how frequently they are spawned and how many spawned simulation objects can exist at one time, is configured in a spawn pattern. VR-Forces comes with two spawn patterns. You can edit spawn patterns and create as many new ones as you wish. Spawn points and sink points for new and edited spawn patterns are dynamically added to the Create menu and Create Object Palette so that it is easy to create new spawn and sink points for all of the available patterns.

The SubwayPatternOfLlife example scenario has spawn patterns that demonstrate continuous simulation object creation, burst traffic, and use of a custom plan.

You can find links to videos that demonstrate spawn patterns at https://www.mak.com/support/tutorials, or click the camera icon next to the procedure whose video you want to view.

Spawn and sink points are automatically added to the Create menu. However, they are not flagged as Favorites on the Create Object Palette. If you flag a spawn or sink point as a Favorite, another entry is added to the Create menu of the specified Force.

16.2.1 Creating Spawn Points

You can create spawn points from the Create menu, the Create Object Palette, or from the Spawn Pattern Manager. Figure 153 illustrates a spawn point.

Figure 153. Spawn point



If the plan for a spawn point specifies Prefer Roads and a spawn point is placed too far away from the road network to use roads, a message is sent to the spawn point's console. If warning icons are enabled, a warning icon displays. For details about the Prefer Roads option, see "Creating a New Spawn Pattern" on page 422. For details about configuring console messages and warning icons, see "21.9. Configuring Object Console Messages" on page 483.

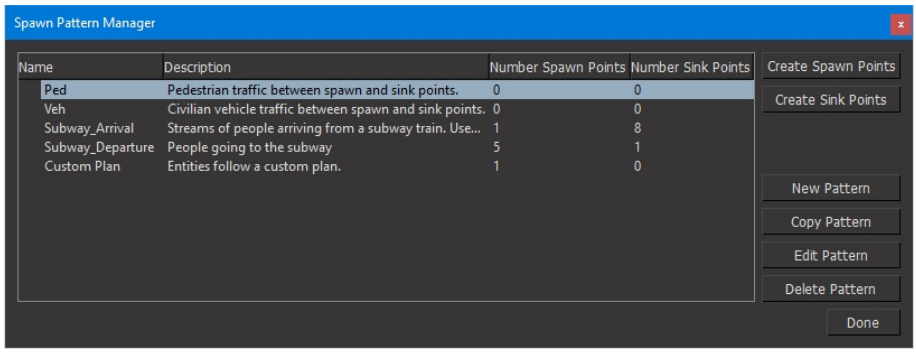
➤ **To create a spawn point from the Create menu:**

1. Choose **Create > Clutter >** *type*, where *type* is a spawn point for one of the spawn patterns. The cursor changes to the spawn point symbol.
2. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
3. Right-click to exit create mode.

➤ **To create a spawn point from the Spawn Pattern Manager:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154).

Figure 154. Spawn Pattern Manager



2. Select the spawn pattern for which you want to create spawn points.

3. Click Create Spawn Points. The cursor changes to the spawn point symbol.
4. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
5. Right-click to exit create mode.

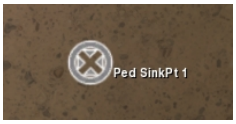
Activating and Deactivating Spawn Points

You can activate and deactivate spawn points. To do so, use the Activation set data request. For details, see "48.2. Activation" on page 1038.

16.2.2 Creating Sink Points

If you are using the default spawned simulation object behavior – pick a sink point and move to it – then you have to create one or more sink points for the simulation objects to move to. Entities created at a given type of spawn point automatically choose among sink points of the same type. (You can configure spawned simulation objects to move to any kind of point by editing the spawn pattern. For details, see "Creating a New Spawn Pattern" on page 422.) You can create sink points from the Create menu, the Create Object Palette, or from the Spawn Pattern Manager. Figure 155 illustrates a sink point.

Figure 155. Sink point



If a sink point is more than 10 meters from a road and simulation objects move towards it with the Prefer Roads criteria, they move to the closest point on the road to the sink point and are then removed from the scenario.

➤ **To create a sink point from the Create menu:**

1. Choose **Create > Clutter >** *type*, where *type* is a sink point for one of the spawn patterns. The cursor changes to the sink point symbol.
2. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
3. Right-click to exit create mode.

➤ **To create a sink point from the Spawn Pattern Manager:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154 on the previous page).
2. Select the spawn pattern for which you want to create sink points.
3. Click Create Sink Points. The cursor changes to the sink point symbol.
4. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
5. Right-click to exit create mode.

16.2.3 Spawn Patterns

A spawn pattern configures these aspects of spawned simulation object creation and behavior:

- The name and description of the spawn pattern.
- The timing of simulation object creation. Entities can be spawned continuously or in bursts.
- The type of simulation object created.
- The spawned simulation object's plan. Entities can follow a default plan or a custom plan.

A spawn pattern's name identifies it on the Create menu and is used to build the label for individual spawn points, for example Ped SpawnPt 1, Ped SpawnPt 2, and so on.

A scenario can have multiple spawn patterns and each spawn pattern can spawn one or more object types. Spawning multiple simulation object types in one pattern is useful when you want heterogeneous traffic. For example, in a stream of pedestrians, you might want to include men, women, and children. In other cases, you might want a specific simulation object type engaging in specific behaviors and would only spawn one simulation object type in that pattern.

If you use the default plan, you can configure simulation object movement:

- Maximum and minimum speed. VR-Forces randomly assigns a speed between the maximum and minimum settings. It will not accept a speed that is greater than the maximum speed for the simulation object in the OPD file.
- Prefer Roads. Enables or disables road-following behavior for ground vehicles. Road following behavior is described in "23.2.4 Vehicle Movement On Roads" on page 503.

When you use the default plan, the Create Spawn Pattern dialog box lists the sink points that this pattern will use.

The SubwayPatternOfLife example scenario has spawn patterns that demonstrate spawning simulation objects continuously and in bursts. It also demonstrates use of a custom plan. The following sections describe these different ways to set up spawn patterns.

How Spawn Events are Scheduled

Spawn patterns control the time period during which simulation objects are spawned and the rate at which they are spawned during that time period. A spawn pattern can use one of these options:

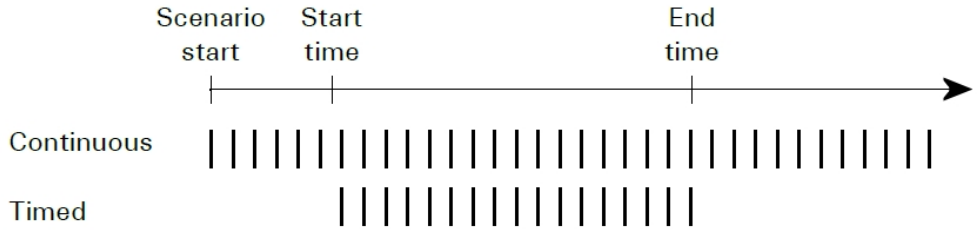


- Continuously. Entities are spawned as soon as the scenario starts and throughout its life.
- Time of Day. Entities are spawned between a starting time of day and ending time of day.

- Simulation Time. Entities are spawned between a specific starting and ending simulation time.

Figure 156 illustrates the difference between continuous and timed spawning.

Figure 156. Continuous spawning compared to time-based

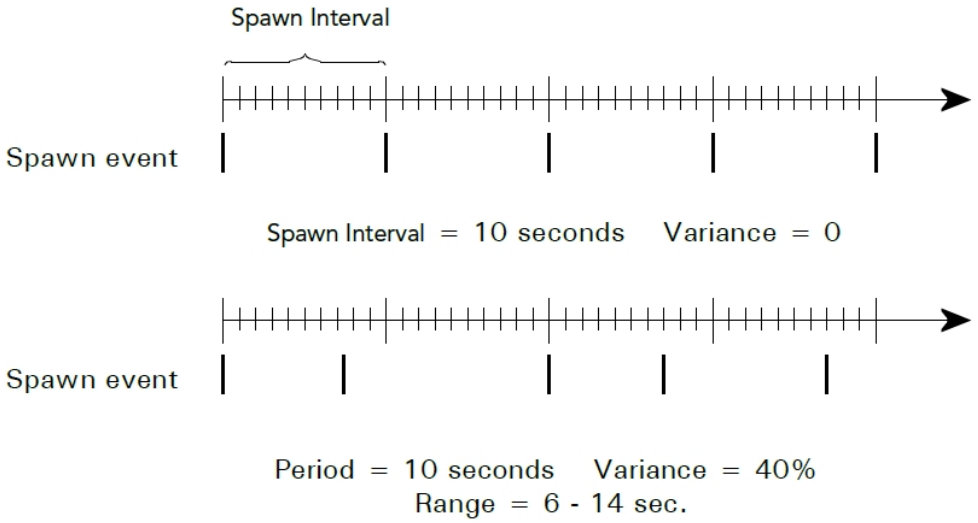


After you specify a time option, you must specify a spawn interval and a spawn interval variance. The spawn interval determines how frequently simulation objects are spawned. The spawn interval variance introduces some variability to the interval so that there is a more random and naturalistic flow of simulation object creation.

For example, if the spawn interval is 10 seconds, a simulation object is spawned every 10 seconds, plus or minus the variance. The variance is a percentage of the spawn interval. If the variance is 10%, then given a spawn interval of 10 seconds, an simulation object is spawned every 9 to 11 seconds. The spawn intervals are not fixed to an absolute time scale. A new spawn interval starts each time a spawn event occurs.

Figure 157 illustrates a series of spawn events with a spawn interval of 10 seconds and a spawn interval variance of 0% and a series of hypothetical events with a spawn interval of 10 seconds and a spawn interval variance of 40%. In the second case, each spawn event can take place anywhere between 6 and 14 seconds after the previous event. Notice that a new spawn interval starts at each spawn event.

Figure 157. Spawn period and variance



The frequency with which a spawn point spawns simulation objects can be affected by the total number of simulation objects that it has spawned. If the maximum number of simultaneous simulation objects for a spawn point is reached, no more simulation objects are spawned from that point until simulation objects from that spawn point are removed from the scenario.

Spawning Entities in Bursts

You can introduce additional variation to the spawn frequency by specifying that simulation objects be spawned in bursts. A burst pattern controls spawning behavior within a fixed period of time. For example, you could use a burst pattern to represent a cycle of train disembarkations or rush hour traffic.

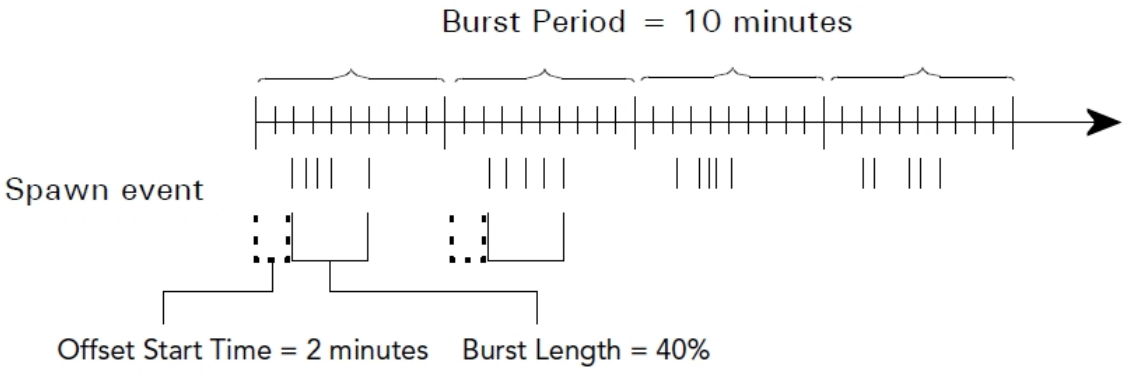
Burst patterns have these parameters:

- Burst Period. The period of time during which a burst can take place.
- Burst Length. The percentage of the burst period in which to spawn simulation objects.
- Offset Start Time. An offset from the start of the burst period at which to start spawning simulation objects.

Within a burst, simulation objects are created at the rate determined by the spawn interval and spawn interval variance values.

Figure 158 illustrates a series of bursts. The burst period is 10 minutes. The offset start time is two minutes, so each burst starts two minutes after the start of the burst period. The burst lasts for 40% of the length of the burst period, in this case four minutes. Within each burst, the frequency of spawn events is controlled by the spawn interval and variance values.

Figure 158. Burst periods



Using a Custom Plan

As noted in "16.2. Spawn and Sink Traffic Generation" on page 416, the default behavior for a spawned simulation object is to choose a sink point, move to the sink point, and be removed from the scenario. However, you can introduce more complex behaviors for spawned simulation objects by writing a custom plan for a spawn pattern. The custom

plan enables the simulation objects to engage in a complex set of behaviors, but you still have the benefit of automatic generation. An example of using a custom plan would be to generate a steady stream of jets taxiing to a runway and taking off from an airport.

Custom plans can include all of the tasks, sets, conditions, and global commands that a regular individual plan has. The plan is treated as an individual plan and applied to each simulation object as it is created. When you write a custom plan, simulation objects do not move to sink points. They remain in the scenario unless their plan removes them. You can delete simulation objects without needing to specify the specific simulation object with the Delete Self global command.

To enable IFF (Identification, Friend or Foe) in aircraft generated by a spawn point, write a custom plan that includes a Set to turn on IFF. Note, however, that in spawn points IFF does not include unique IFF identification codes or realistic flight IDs.

Default Spawn Patterns and Scenario-Specific Patterns

The VR-Forces includes default spawn patterns for civilian vehicles and civilian lifeforms. When you create a scenario, these patterns are available in the scenario. When you save a scenario, the spawn patterns are saved with the scenario in the scenario extras file (*.xtr*). (This includes the default spawn patterns if you do not edit them, the edited versions if you have edited them, and any new spawn patterns you have created.) They are now scenario-specific. The next time you load the scenario, it uses the spawn patterns in the scenario extras file, not the default spawn patterns.

If you want to have additional default spawn patterns for your scenarios, you can add entries to *./appData/settings/vrfSim/defaultSpawnTemplates.mtl*. Copy one of the patterns in the file, then change the name and parameter values to suit your needs. Or, create a spawn pattern in a scenario and copy it from the *.xtr* file to *defaultSpawnTemplates.mtl*.

> The civilian vehicle spawn pattern included with VR-Forces is set to user roads, therefore it requires a nearby road network to successfully spawn vehicles.

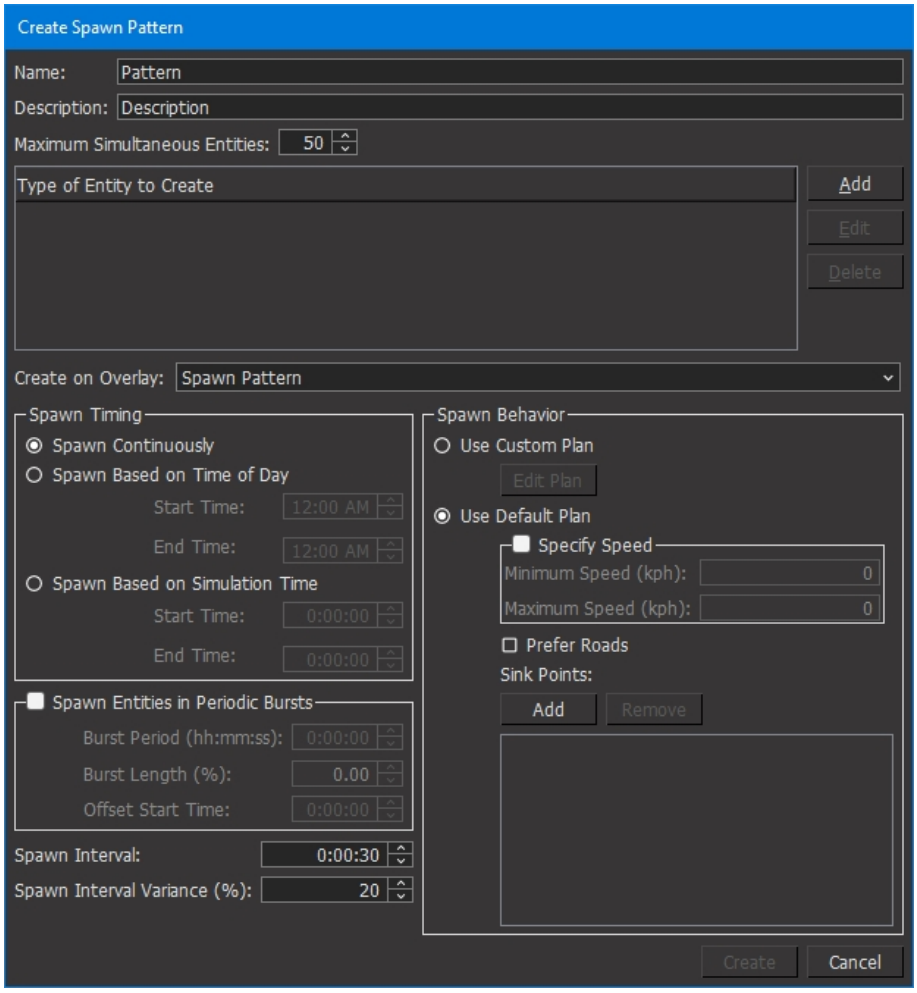
16.2.4 Spawn Pattern Management

Creating a New Spawn Pattern

➤ **To create a spawn pattern:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154 on page 417).
2. Click New Pattern. The Create Spawn Pattern dialog box opens (Figure 159).

Figure 159. Create Spawn Pattern dialog box



3. In the Name box, type a name for the spawn pattern. This name is used for the label for spawn points and sink points created from this pattern. It is a required field.
4. In the Description box, type a description for the pattern. This description is listed in the Spawn Pattern Manager to help identify the purpose of the pattern. This is a required field.
5. Specify the maximum number of simulation objects that can be spawned from a spawn point using this pattern and exist in the scenario at the same time. When the maximum is reached for a given spawn point, no more simulation objects are spawned from that point until a simulation object that was spawned from that point is removed from the scenario.

   For example, if the maximum number of simultaneous simulation objects for a spawn pattern is 200 and you have 10 spawn points of that type, you could have as many as 2000 spawned simulation objects. None of those spawn points will individually spawn more than 200 simulation objects at one time.

> ⚠ A high maximum number of simultaneous simulation objects along with a small spawn interval can quickly generate tens to hundreds of simulation objects. This can affect performance.

6. Specify the simulation object types to create:
    a. Click the Add button. The Object Type dialog box opens.
    b. Optionally, filter the object list. (For information about how to filter an objects list using tags, see "14.2.3 Filtering the Create Object Palette" on page 377.)
    c. Select the simulation object type that you want to add. The simulation object is added to the Type of Entity to Create list.
    d. Add additional simulation object types as desired.

> ⓘ Using the Object Type dialog box, you can specify all types of simulation objects, cultural features, and control objects, some of which do not necessarily make any sense to spawn. It is up to you to select types of simulation objects that make sense.

7. In the Create on Overlay list, select the overlay on which you want spawned simulation objects to be created.
8. Choose an option in the Spawn Timing group box:
    — Spawn Continuously. Entities are spawned throughout the simulation.
    — Spawn Based on Time of Day. Specify the starting time of day and ending time of day during which you want simulation objects to be spawned.
    — Spawn Based on Simulation Time. Specify the starting simulation time and ending simulation time during which you want simulation objects to be spawned.
9. Optionally, select Spawn Entities in Periodic Bursts. (For an explanation of periodic bursts, see "Spawning Entities in Bursts" on page 421.) If you select it, configure these values:
    — Burst Period. The time period for each burst.
    — Burst Length. The length of time, as a percentage of the burst period, during which simulation objects are spawned.
    — Offset Start Time. An offset from the start of the burst period to start spawning simulation objects.
10. Specify the spawn interval. This specifies the amount of time between each spawn event, modified by the spawn interval variance.
11. Specify the spawn interval variance as a percentage of the spawn interval.

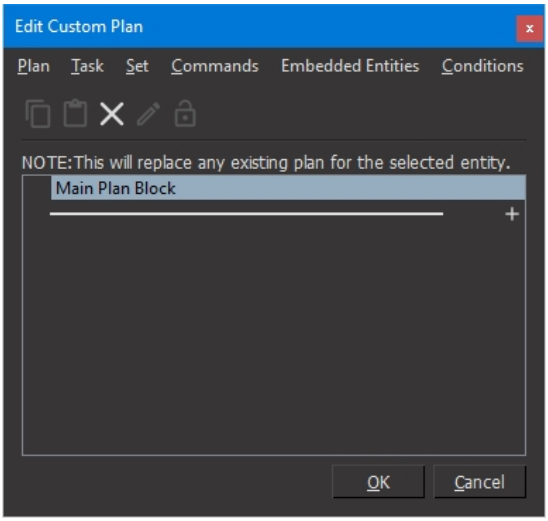
12. In the Spawn Behavior group box, select a behavior option:

    To use Custom Plan:

    a. Click Edit Plan. The Edit Custom Plan dialog box opens.

    b. Write a plan that each simulation object created from this spawn pattern will use. Unless you want the simulation objects to remain in the scenario permanently, end the plan with a Delete Self command. (For details about how to write plans, see "Chapter 43. Writing Plans" on page 947.)

Figure 160. Edit Custom Plan dialog box



    To use a Default Plan:

    a. Optionally, select the Specify Speed check box and specify the minimum and maximum speeds for spawned simulation objects. If you do not select this option, spawned simulation objects move at the default ordered speed in their OPD file. If you select it, they move at randomly chosen speeds between the minimum and maximum values. This option introduces some variance in behavior. Although the GUI allows you to set any maximum speed, simulation objects will not move faster than the maximum speed in the OPD file.

    b. To have a simulation object move on roads, select the Prefer Roads check box. If this option is cleared, simulation objects move directly towards sink points without regard to vector road networks.

> ⚠ Prefer Roads is valid only for ground vehicles. Do not select this option for any other simulation object type.
>
> If Prefer Roads is selected, spawn points must be within 10 meters of the road. If a spawn point is farther away, the simulation objects are removed from the scenario as soon as they are created.
>
> If a sink point is more than 10 meters from a road, simulation objects that are moving towards that sink point move to the point on the road that is closest to it and are then removed from the scenario.

c. Optionally, use the Add button to edit the Sink Points list. A new spawn pattern will not have any sink points listed because until you create the spawn pattern, you cannot create any sink points for that pattern. Entities will automatically use sink points that match the spawn pattern, so after you create them, you do not have to add them to the list. However, they can also move to waypoints. If you want spawned simulation objects to move to waypoints, you must add them to the list.

13. Click Create.

Copying a Spawn Pattern

If you want to create a new spawn pattern that will be similar to an existing spawn pattern, you can copy the existing spawn pattern to a new name.

➤ **To copy a spawn pattern:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154 on page 417).
2. Select the spawn pattern that you want to copy.
3. Click Copy Pattern. The Create Spawn Pattern dialog box opens. It has a default name based on the original pattern name.
4. Edit the spawn pattern. The meaning of each option is described in "Creating a New Spawn Pattern" on page 422.
5. Click Create.

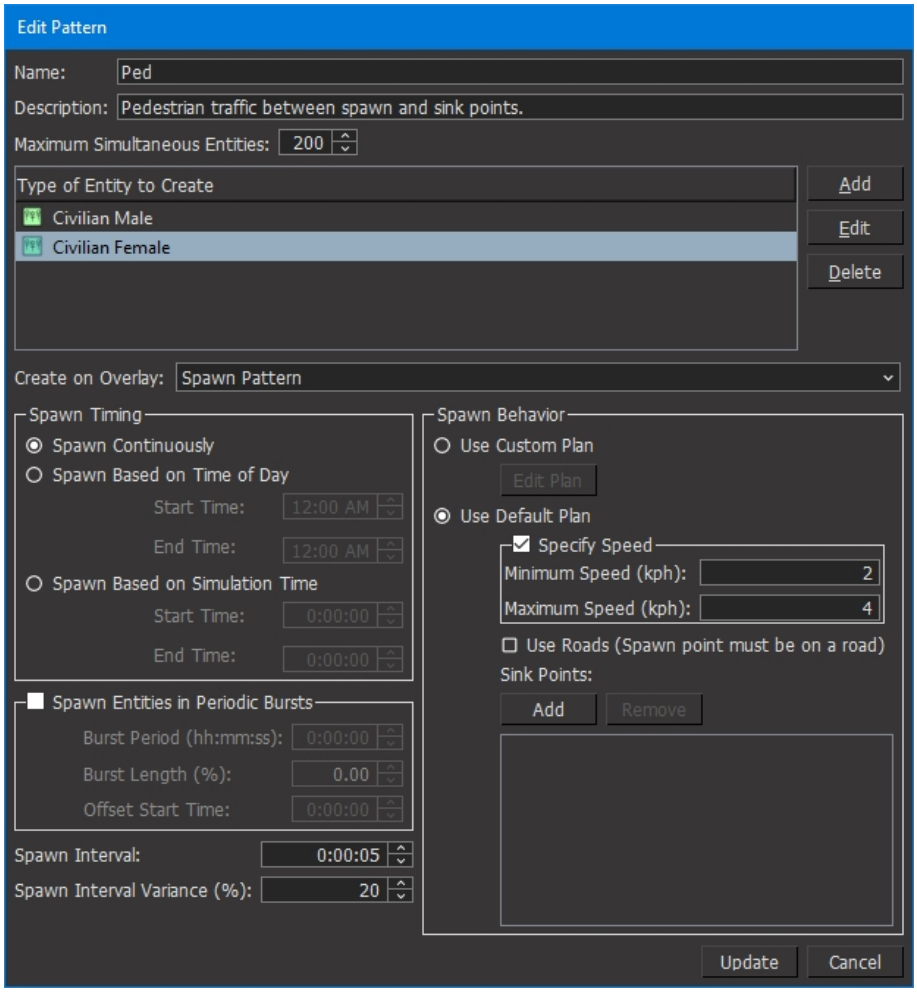
Editing a Spawn Pattern

You can change any of the parameters of a spawn pattern. The changes do not affect any simulation object that has already been spawned.

➤ **To edit a spawn pattern:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154 on page 417).
2. Select the pattern that you want to edit.
3. Click Edit Pattern. The Edit Spawn Pattern dialog box opens (Figure 161).

Figure 161. Edit Spawn Pattern dialog box



4. Change any value as desired. The meaning of each option is described in "Creating a New Spawn Pattern" on page 422.
5. Click Update.

Deleting a Spawn Pattern

If you delete a spawn pattern, all spawn points of that type are deleted from the scenario. Sink points are not deleted. The spawn and sink points of this type are removed from the Create menu and Create Object Palette. If the scenario is running, any simulation objects that were spawned from these points continue executing their plans.

➤ **To delete a spawn pattern:**

1. Choose **Simulation > Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 154 on page 417).
2. Select the spawn pattern that you want to delete.
3. Click Delete Pattern. You are prompted to confirm the deletion.
4. Click Yes.

16.3. Clutter Areas

Clutter areas allow you to use automatic identification system (AIS) or automatic dependent surveillance-broadcast (ADS-B) data included with VR-Forces to create water or air traffic as "clutter objects". This provides you with realistic background clutter, which is useful when you are simulating the real world and you need realistic "noise" on the water or in the air. The clutter objects are all of the same force as the area.

- The prerecorded AIS and ADS-B data were acquired and then preprocessed by MAK. You cannot use your own data.
- There are forty-eight hours' worth of data in the files. If your scenario runs longer than two days, you might experience problems with the clutter objects. If you require more than two days of simulation time, please contact your MAK salesperson or info@mak.com.

Clutter objects are intended to be background entities and display in the Clutter Objects overlay by default. They are not "full-fledged" entities: they are simpler and therefore have a lower impact on performance. When generating clutter objects from AIS or ADS-B data, VR-Forces uses the timestamp in the file to place the clutter objects where they were at the same time of day as the scenario.

Objects move according to the data in the file. If they move out of the clutter area, the object is removed from the scenario. Likewise, if there is data of a ship or fixed-wing aircraft moving into the area, it is added to the scenario.

You cannot assign tasks to clutter objects; they simply move according to the information in the AIS or ADS-B data. If you want to task a clutter object to do something other than what is in the data, use the Promote From Clutter set data request to promote them to a full-fledged entity. After promoting them, you can task the entity as desired. See "40.61. Promote From Clutter" on page 893.

You can customize the clutter entities. See "Clutter Entities" in the *Adding Content to MAK ONE Applications Reference Manual*.

16.3.1 Aircraft Clutter Area

When you create an Aircraft Clutter Area, VR-Forces populates it with fixed-wing aircraft objects based on two days' worth of ADS-B data provided by MAK. VR-Forces uses the data to name the clutter aircraft for the fixed-wings they represent, select a model that most closely matches the type of aircraft, and determine their movement. The aircraft clutter area only creates background fixed-wing aircraft. It does not create rotary wing or other types of aircraft.

The aircraft clutter area does not simulate takeoffs and landings. It only displays fixed-wing aircraft objects that are in the air. When the data has an aircraft take off, the object is added to the scenario. Likewise, when an aircraft lands, the object is removed from the scenario.

Each entity's IFF data displays in the IFF page of the Information dialog box and is published to the network.

➤ **To create aircraft using a clutter area:**

1. Open the Create Object Palette.
2. Optionally, click the Clutter filter ( ).
3. Select Aircraft Clutter Area.
4. Draw an area on the terrain as you would any other areal tactical graphic. When the area is completed, VR-Forces creates objects to populate the area.

16.3.2 Ship Clutter Area

When you create a Ship Clutter Area, VR-Forces populates it with watercraft objects based on two days' worth of AIS data provided by MAK. VR-Forces uses the data to name the clutter ships for the watercraft they represent, select a model that most closely matches the type of vessel, and determine their movement.

Each entity's AIS data displays in the AIS page of the Information dialog box.

➤ **To create ships using a clutter area:**

1. Open the Create Object Palette.
2. Optionally, click the Clutter filter ( ).
3. Select Ship Clutter Area.
4. Draw an area on the terrain as you would any other areal tactical graphic. When the area is completed, VR-Forces creates objects to populate the area.



17. Crowds and Herds

This chapter explains how to create and task pedestrian crowds and how to create animal herds.

17.1. Crowds and Herds in Scenarios 432
17.2. Crowd and Herd Units 432
17.2.1 Creating Crowd Units 432
17.2.2 Creating Herd Units 433
17.3. Crowd and Herd Areas 434
17.3.1 Creating Crowds Using a Pedestrian Area 434
17.3.2 Creating Animal Herds Using Areas 434
17.3.3 Adding Entities to a Pedestrian or Herd Area 435
17.4. Crowd Tasks 435
17.5. Pedestrian-Oriented Tactical Graphics 435
17.5.1 Using Civilian Visit Points 436

17.1. Crowds and Herds in Scenarios

VR-Forces can quickly create multiple simulation objects, which allows you to create crowds of people and herds of animals without placing each simulation object by hand. You can use any of the following methods to create multiple objects:

- Crowd unit. Create a crowd unit from the Create Object Palette or aggregate individual humans into a crowd unit. Assign crowd-specific tasks to the crowd.
- Herd units. Create a herd unit of cows, goats, or sheep from the Create Object Palette. They graze and wander around the central point where you placed the unit.
- Pedestrian area. A pedestrian area is populated by a random number of male and female civilians. They execute a wander task.
- Herd areas. An area is populated by a random number of cows, goats, or sheep. The herd animals graze and wander around the area.
- You can also use spawn points or clutter areas to create a number of entities. See "Chapter 16. Generating Traffic (Pattern of Life)" on page 415

Lifeforms require a navigation area because they perform movements such as the Wander task. You must place two-legged lifeforms (humans and chickens) in a lifeform nav. area and four-legged lifeforms (cows, sheep, goats, and so on) in a ground platform nav. area. If you try to place lifeforms in a location that does not have a nav. area, VR-Forces displays a warning. See "Chapter 66. Generating Navigation Data" on page 1271 to learn more.

17.2.  Crowd and Herd Units

You can create a crowd or herd by placing a unit of the type you need. VR-Forces then creates entities of the particular type.

Based on the entity file, the entities may have random appearances. You cannot control the entities' appearances when you create the unit, but you can set the available appearances for the entity in the Simulation Object Editor. For more information, see "70.1.10 Editing a Lifeform's Visual Model and Animation" on page 1383.

17.2.1 Creating Crowd Units

VR-Forces has six preconfigured crowd units: small, medium, and large crowds of western or middle eastern civilians. They create a unit of civilian males and females. You can also create crowd units by aggregating individual human entities. However, some of the animations used in crowd tasks are specific to the civilian characters used in the supplied units.

When you place a crowd, if the terrain has pedestrian paths or other pedestrian-specific locations, the members of the crowd are placed in these locations.

Crowd members are created within a configurable radius of the location you specify, with a configurable density. By default, crowds are created using the parameters in Table 23. You can change the values in the Simulation Object Editor.

Table 23. Crowd units

Unit	Radius (m)	Density	Approximate Size*
Civilian Crowd (Small)	15	2	50
Civilian Crowd (Medium)	30	2	100
Civilian Crowd (Large)	50	2	175
Civilian Mideast Crowd (Small)	15	2	50
Civilian Mideast Crowd (Medium)	30	2	100
Civilian Mideast Crowd (Large)	50	2	175
*If there are not enough valid locations to place entities, the size of the unit may be smaller than the theoretical maximum.			

Crowd units are supported only in entity-level scenarios.

> ⚠ You can create preconfigured crowds only in navigation areas and crowd tasks can be executed only in navigation areas. Although you can aggregate individual entities into a crowd outside of a navigation area, they will not be able to execute any crowd tasks.

➤ **To create a crowd:**

1. Open the Create Object Palette.
2. Click the Entity filter ().
3. Clear any tag filters (click the X button on the Filters line).
4. Optionally, click the Human top-level tag, then the Civilian tag, and then click the Crowd tag.
5. Select the crowd unit you want.
6. Click on the terrain to place the unit.

17.2.2 Creating Herd Units

VR-Forces has three preconfigured herd units: cows, goats, and sheep. Each unit generates thirty-six animals that graze and wander within an approximate 20-meter radius around the point where you placed the unit. You can also create herd units by aggregating individual animals, but they will not wander automatically.

➤ **To create a herd:**

1. Open the Create Object Palette.
2. Click the Entity filter ().
3. Clear any tag filters (click the X button on the Filters line).

4. Optionally, click the Animal top-level tag and then click the Crowd tag.
5. Select the herd unit you want.

17.3.  Crowd and Herd Areas

You can draw areas to create crowds of people or herds of animals. VR-Forces then creates entities of the particular type.

Based on the entity file, the entities may have random appearances. You cannot control the entities' appearances when you create the area, but you can set the available appearances for the entity in the Simulation Object Editor. For more information, see "70.1.10 Editing a Lifeform's Visual Model and Animation" on page 1383.

17.3.1 Creating Crowds Using a Pedestrian Area

When you create a pedestrian area, VR-Forces populates it with a random number of male and female civilian entities. They are created as a ground unit. They are all assigned a wander task. You cannot specify the number of entities to create or the mix of entities to create when you create the area.

> ⚠ If you delete a pedestrian area, all of the entities that were created for it are also deleted.
>
> You can create pedestrian areas only in navigation areas.

➤ **To create a crowd using a pedestrian area:**

1. Open the Create Object Palette.
2. Optionally, click the Clutter filter ().
3. Select Pedestrian Area.
4. Draw an area on the terrain as you would any other areal tactical graphic. When the area is completed, VR-Forces creates entities to populate the area.

17.3.2 Creating Animal Herds Using Areas

You can create herds of cows, goats, or sheep using the appropriate area type. Once you create the area, VR-Forces populates it with a random number of that type of animal and they will graze and wander within the area. You cannot specify the number of animals to create at first.

➤ **To create a herd using an area.**

1. Open the Create Object Palette.
2. Optionally, click the Clutter filter ().
3. Select the area for the type of herd you want to create.

4. Draw an area on the terrain as you would any other areal tactical graphic. When the area is completed, VR-Forces creates animals of the selected type to populate the area.

17.3.3 Adding Entities to a Pedestrian or Herd Area

After you create a pedestrian or herd area, you can add additional entities to the area. See "32.8. Create More Entities" on page 652.

17.4. Crowd Tasks

Crowds can execute some of the tasks available to other ground units. Crowds created as a unit or in an area can perform these tasks. They have the following movement tasks:

- Circle around a location
- Crowd along a line
- Crowd around a location
- Crowd around an object
- Crowd around waypoints
- Gather in front of an entity
- Wander
- Flee from entities
- Protest along a line
- Protest around a location
- Riot
- Disperse crowd

When a crowd gathers around a location or object, VR-Forces calculates a target destination for each member of the crowd and each member of the crowd tries to move to that destination. If you send two or more crowds to the same location or object, these target destinations may overlap. Individual entities may give up on their movement tasks or may end up very close to members of the other crowd.

For details about each task, see "Chapter 32. Human and Crowd Tasks" on page 649.

17.5. Pedestrian-Oriented Tactical Graphics

VR-Forces has tactical graphics that are particular to pedestrians. These objects only work in navigation areas.

- Pedestrian path. When humans plan paths in navigation areas that have navigation data, if a pedestrian path is present that can take them to their destination, they will use it. You can think of a pedestrian path as a route that a human automatically

chooses to follow instead of being told to follow it. Pedestrian paths are supported only for entity-level scenarios.

- Civilian visit points. Waypoints that attract the attention of civilians.
- Crosswalks. Crosswalks are similar to pedestrian paths. Pedestrians prefer them when path planning. The unique aspect of crosswalks is that if an entity is configured with a human crosswalk controller, it checks to see if it is safe to cross the street before proceeding. (Crosswalks are an experimental feature. Please contact support (https://www.mak.com/mak-one/support/portal) if they do not work as you expect.)
- Pedestrian areas. Pedestrian areas automatically generate pedestrians when you create them. For details, see "17.3.1 Creating Crowds Using a Pedestrian Area" on page 434 .

17.5.1 Using Civilian Visit Points

Civilian visit points are waypoints that attract the attention of civilian entities that are walking nearby. The civilians can be members of a crowd or individual entities. When a civilian comes within a configurable distance of a civilian visit point, it stops and faces the point a configurable percentage of the time. If it stops, it waits a few seconds and then continues its task.

This behavior is controlled by the Visit Interest Points reactive task. You can edit the parameters that control an entity's behavior in the Manage Reactive Tasks dialog box. For more information about reactive tasks, see "29.5. Reactive Tasks" on page 572.



18. Creating and Controlling Units

This chapter explains how to create and manage units.

18.1. Creating Units 438
18.1.1 Creating a Unit by Combining Existing Simulation Objects 438
18.1.2 Creating a Preconfigured Unit 440
18.2. Selecting a Unit 440
18.3. Adding Simulation Objects to a Unit 440
18.4. Removing a Simulation Object from a Unit 441
18.5. Deleting a Unit 442

18.1. Creating Units

You can create a unit by selecting simulation objects (which can be individual entities or units) and combining them to form a unit, or by placing a preconfigured unit from the Create Object Palette.

When you create a unit, it is created as a subordinate to the force level. You cannot create units that are subordinates of an existing unit. Once you create a unit, you can subordinate it to another unit. (For details, see "18.3. Adding Simulation Objects to a Unit" on page 440).

> Except where noted, the procedures for creating and editing units apply to entity-level scenarios and aggregate-level scenarios.

18.1.1 Creating a Unit by Combining Existing Simulation Objects

When you create a unit by combining simulation objects, you can specify the order of subordinates in the unit. This determines which subordinate is considered the unit leader and affects the assignment of echelon IDs. (It also affects the icon used to represent the unit.) You cannot change the subordinate order after you create the unit.

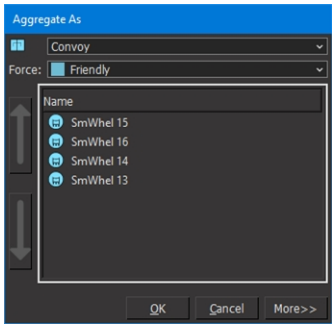
VR-Forces has many unit types configured in the Simulation Object Editor that are not available to create from the Create Object Palette, but are available as options on the Aggregate As dialog box.

- When you create a unit, the members of the unit get new echelon IDs to reflect the new entity hierarchy.
- When you aggregate individual entities into a unit, you can include non-VR-Forces entities. These entities will be included for the purposes of calculating the bounding volume of the unit and for expanding and collapsing the display of unit members. However, VR-Forces actions that would not typically apply to non-VR-Forces entities, such as tasks and set data requests, will not affect the non-VR-Forces entities.

➤ **To create a unit by combining existing simulation objects:**

1. Select the simulation objects that you want to become part of a unit.
2. Choose **Objects > Unit > Aggregate As**. The Aggregate As dialog box opens (Figure 162).

Figure 162. Aggregate As dialog box



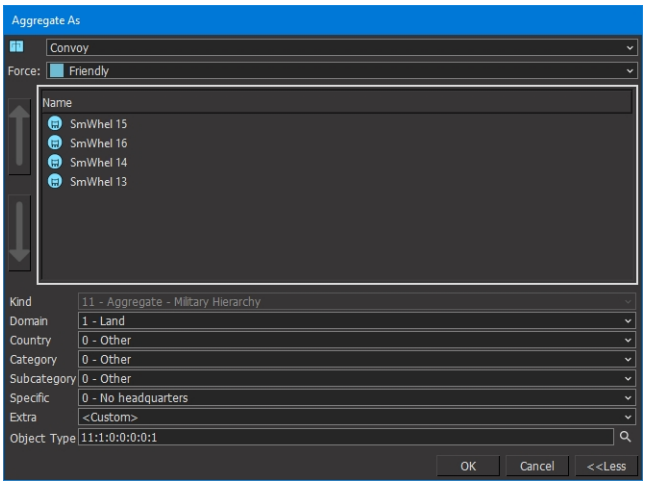
3. Select a unit type from the list.



- The Convoy unit can only be assigned convoy tasks. Its only purpose is to allow you to quickly aggregate miscellaneous ground vehicles into a convoy.
- The default list of unit types is limited and is different for entity-level scenarios and aggregate-level scenarios. You can specify the unit types that are on the list in the Simulation Object Editor. For details, see "71.3. Specifying an Aggregate As Option" on page 1406.

4. Optionally, customize the entity type enumeration for the unit:
   a. Click More. The dialog box expands to show the entity type enumeration for the unit type selected (Figure 163).

Figure 163. Aggregate As dialog box



   b. Edit the enumeration by selecting values from the lists, or by typing an enumeration.

> If an edit to the enumeration results in an entity type that does not match the unit type selected in the list, the list entry does not change. However, the unit will be created with the entity type specified by the custom enumeration.

5. Optionally, change the order of subordinates:
   a. Select the subordinate whose order you want to change.
   b. Click the up or down arrow to move it to a new position in the list of subordinates.
6. Click OK.



Creating Higher Echelon Units in Aggregate-Level Scenarios

In aggregate-level scenarios, you can combine entities and units to create higher echelon units just as you can in entity-level scenarios.

18.1.2 Creating a Preconfigured Unit

EntityLevel.sms and *AggregateLevel.sms* include preconfigured units at several hierarchical levels. You can configure them and add additional units in the Simulation Object Editor. (For details, see "71.4. Editing Unit Composition" on page 1407.)

You create preconfigured units by selecting them on the Create Object Palette and follow the standard procedures for creating simulation objects. The abbreviations in the unit names have the following meanings:

- ADA. Air defense artillery.
- COLT. Combat observation lasing team.
- CSS. Combat service support.
- FA. Field artillery.
- MI. Mechanized infantry.

18.2. Selecting a Unit

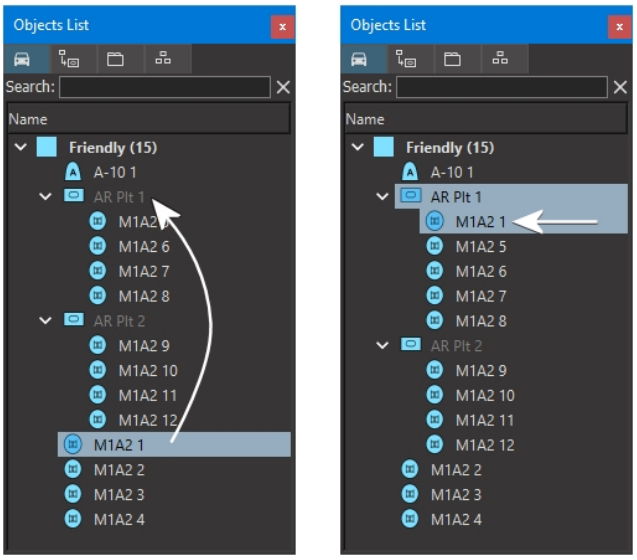
➤ To select a unit, select its icon on the terrain, or select it in one of the views on the Objects List Panel.

18.3. Adding Simulation Objects to a Unit

➤ To add a simulation object to a unit, in the Objects List View, select the simulation object you want to add and drag it onto the unit as illustrated in Figure 164. The name of the simulation object stays the same, but its echelon ID changes. (You can see the echelon ID in the object's Information dialog box.)

> You can add simulation objects to a unit with the Superior set data request. For details, see "40.80. Superior" on page 902.

Figure 164. Adding a simulation object to a unit

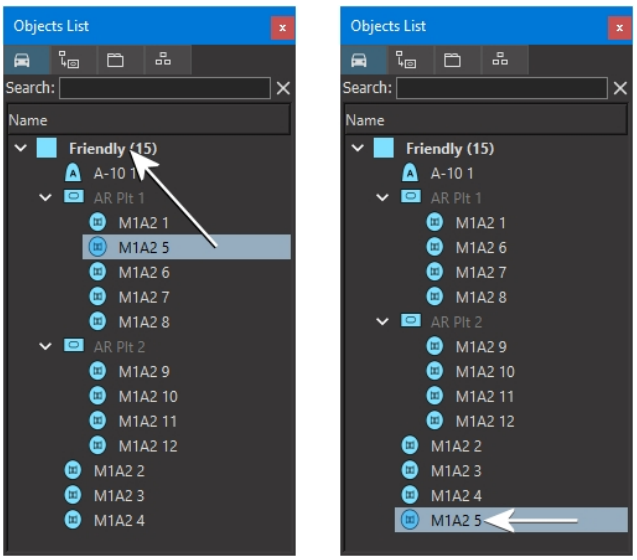


18.4. Removing a Simulation Object from a Unit

➤ To remove a simulation object from a unit, in the Objects List View, drag the simulation object from the unit to the force level in the window, as illustrated in Figure 165, or into another unit. The name of the simulation object stays the same, but its echelon ID changes. (You can see the echelon ID in the object's Information dialog box.)

You can remove simulation objects from a unit with the Superior set data request. For details, see "40.80. Superior" on page 902.

Figure 165. Removing a simulation object from a unit



18.5. Deleting a Unit

⚠ If you delete a unit, all of its members are deleted.

➤ **To delete a unit:**

1. Select the unit.
2. Choose **Objects > Delete**.



19. Displaying Units

Many aspects of managing units are the same as for individual entities. This chapter focuses on features that are unique to units.

19.1. Selecting Units 444
19.2. Expanding and Collapsing Units 444
19.2.1 Expanding or Collapsing a Single Unit 445
19.2.2 Expanding or Collapsing All Units 445
19.3. Displaying Ghosted Icons 446
19.4. Displaying Unit Icons and Bounding Volumes 448
19.4.1 Specifying the Color Scheme for Unit Bounding Volumes 450

19.1. Selecting Units

You can select units in the same ways that you can select entities. (For details about selecting entities, see "20.2. Selecting Simulation Objects, Tactical Graphics, and Props" on page 453.) However, to select a unit, you must select the unit icon, not individual members of the unit. If a unit is expanded, the unit icon might not be visible. To select the unit, you can collapse it, or you can select the unit in the Objects List View.

19.2. Expanding and Collapsing Units

You can display units in expanded or collapsed mode. The expanded or collapsed state is saved as part of a scenario.

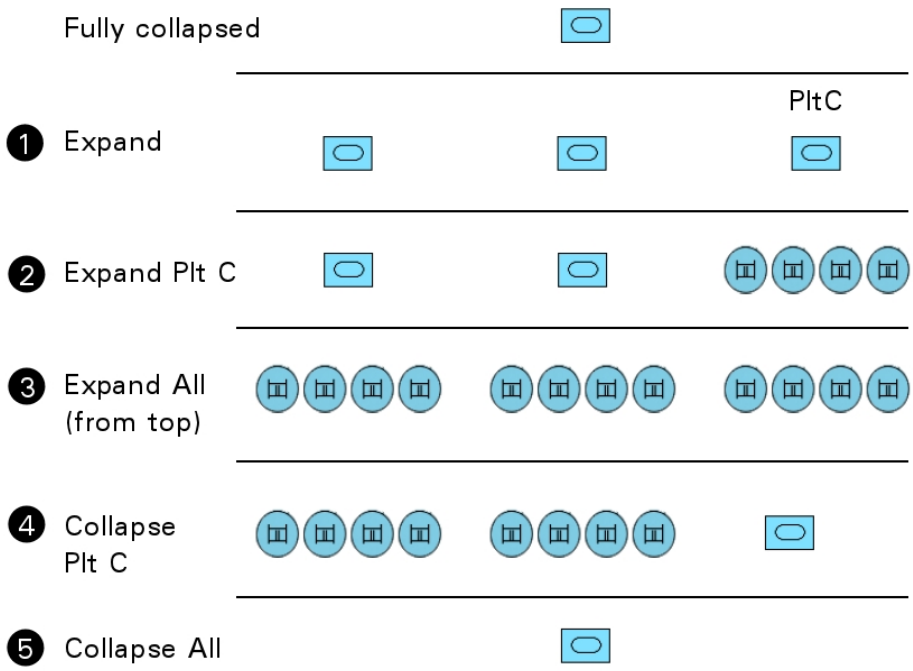
When a unit is collapsed, it displays as one icon. When a unit is expanded, all the members of the unit are visible as individual entities or subordinate units and the unit icon is hidden. Multi-level units can expand and collapse at each level of aggregation. For example, a company could be collapsed to the company level (one icon), the platoon level (an icon for each platoon), or fully expanded.

When you Expand or Collapse a unit, you can expand or collapse one level of aggregation or all levels. When you Expand or Expand All, expansion is downward from the current unit and does not affect any sibling units. When you Collapse, contraction is upward in the hierarchy. Collapse All collapses all members of the unit to the topmost level.

Figure 166 illustrates these relationships. Assume a company with three platoons, each of which has four vehicles.

- If you Expand the top level unit, it expands to the platoon level. (1)
- If you select Plt C and choose Expand or Expand All, it expands just Plt C. (2)
- If you select the top level unit and expand all, it expands to the leaf level – 12 individual vehicles. (3)
- If the unit is fully expanded and you select one member of Plt C and choose Collapse, just Plt C collapses. (4)
- If you select any member of an expanded unit and select Collapse All, the entire unit collapses. (5)

Figure 166. Unit expand and collapse



19.2.1 Expanding or Collapsing a Single Unit

➤ **To expand a unit:**

1. Select the unit icon or its entry in the Objects List View of the Objects List Panel.
2. Choose **Objects > Unit > Expand**.

➤ **To expand a unit and all its subordinate units:**

1. Select the unit icon or its entry in the Objects List View of the Objects List Panel.
2. Choose **Objects > Unit > Expand All**.

➤ **To collapse a unit one level:**

1. Select any member of the unit.
2. Choose **Objects > Unit > Collapse**.

➤ **To collapse a unit to its root level:**

1. Select any member of the unit.
2. Choose **Objects > Unit > Collapse All**.

19.2.2 Expanding or Collapsing All Units

Using the Aggregate Display Toolbar, you can expand or collapse all units in the scenario.

➤ To collapse all units to their root levels, click the Collapse All button ( ).

➤ To expand all units and their subordinates, click the Expand All button ( ).

The Collapse All and Expand All buttons apply only to units that are known at the time you click the button. Any units added to the scenario after you click the button display in their default state.

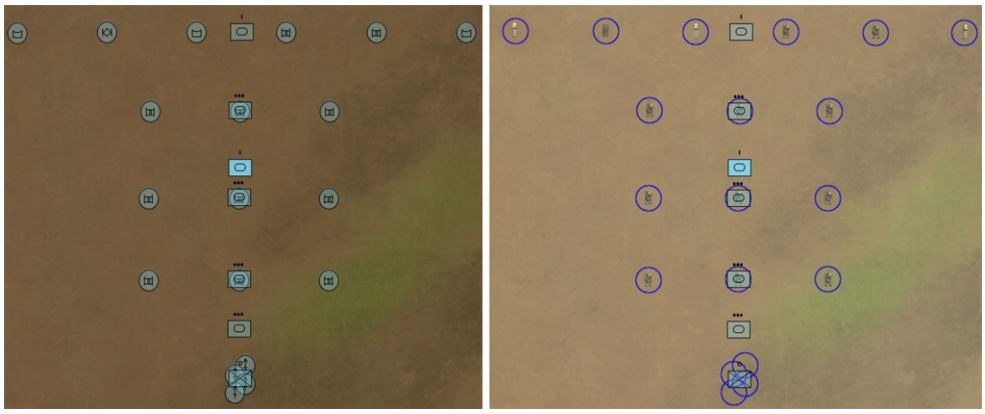
19.3. Displaying Ghosted Icons

If you want to select units, you need to collapse them so that you can select the unit icon. However, when you do this, you can no longer see where individual subordinate simulation objects are located and if subordinates are destroyed, you cannot see which ones are destroyed and which are functional. VR-Forces provides the following options for displaying unit icons to allow you to work with them in a way that best meets your visualization and simulation object management needs:

- Display ghosted icons for hidden simulation objects.
- Allow ghosted icons to be selected.
- Display all unit members.
- Manually expand and collapse the display.
- Display leaf-level unit members.

In the Plan View observer mode, when you display ghosted units, the unit icon displays normally, and the subordinate members are displayed using semi-transparent icons. In 3D observer modes, units that are subordinate to the top level unit are displayed using semi-transparent icons. Leaf-level entities are displayed using the appropriate model for the observer mode (Figure 167).

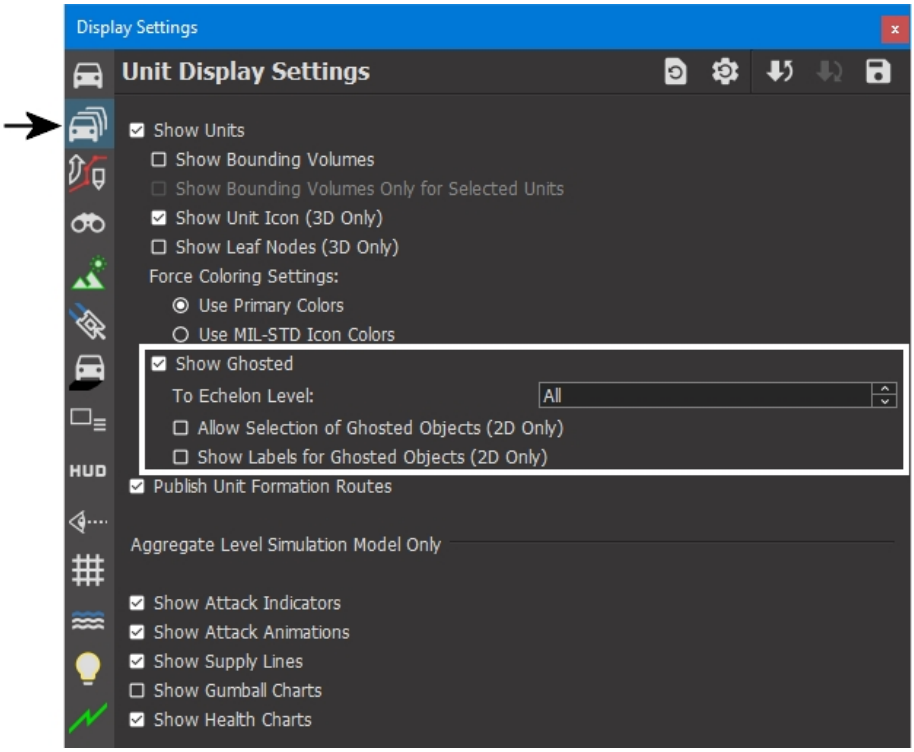
Figure 167. Ghosted icons - 2D and 3D



➤ **To configure the display of unit icons in Unit Display Settings:**

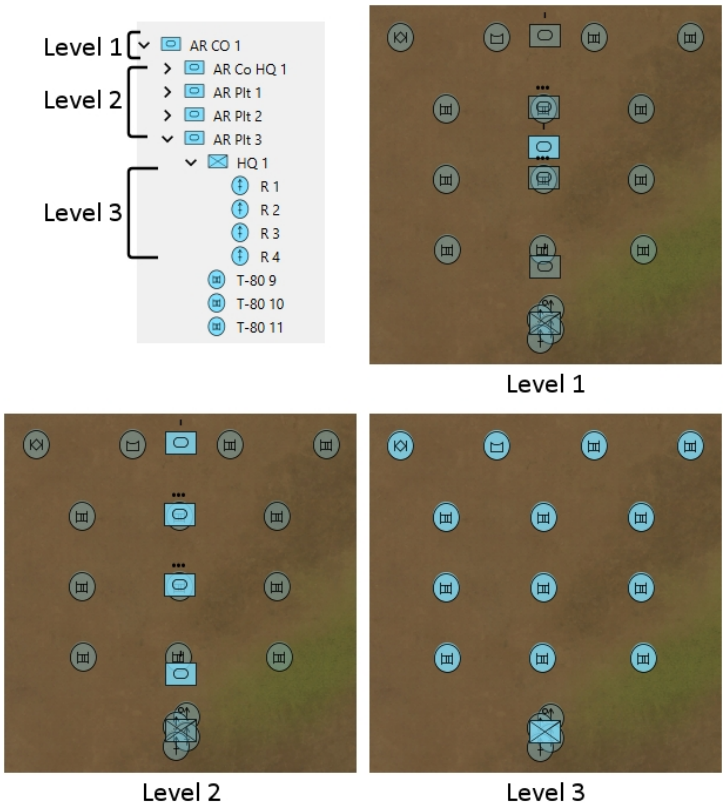
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 168).

Figure 168. Unit Display Settings page



3. To display unit members based on the current level of aggregation only, clear Show Ghosted.
4. To display ghosted icons:
    a. Select Show Ghosted.
    b. In the To Echelon Level box, select the level of aggregation to show ghosted icons for. To show all levels without needing to consider how many levels there are, select All. Figure 169 illustrates the effect of changing the echelon level for displaying ghosted icons for the VR-Forces unit hierarchy shown.
    c. To allow selection of ghosted icons (2D view only), select Allow Selection of Ghosted Icons. (In 3D views, you can always select entities at the leaf level.)
    d. To display symbol decorations for ghosted icons (2D view only), select Show Labels for Ghosted Objects.

Figure 169. Displaying ghosted icons by echelon level



➤ **To display ghosted icons using the Aggregate Display Toolbar:**

1. Enable the Show/Hide Ghosted icon to display ghosted icons ( ).
2. Use the arrow to access the slider and adjust the level of aggregation to show ghosted icons for.

19.4. Displaying Unit Icons and Bounding Volumes

VR-Forces lets you configure when to display unit icons and bounding volumes. Bounding volumes are translucent polygons that show the extents of the unit. This is useful when the individual entity icons or models are not displayed. You can configure icons and bounding volumes:

- Enable and disable the display of units. When disabled, only the 2D icons or 3D models for leaf level unit members are displayed.
- Enable and disable the display of bounding volumes. Bounding volumes are displayed only for collapsed units. You can restrict the display of bounding volumes to the selected units.
- In 3D observer modes, enable or disable the display of unit icons.

Figure 170 shows a unit bounding volume in Stealth observer mode, with leaf nodes displayed.

Figure 170. 3D bounding volume

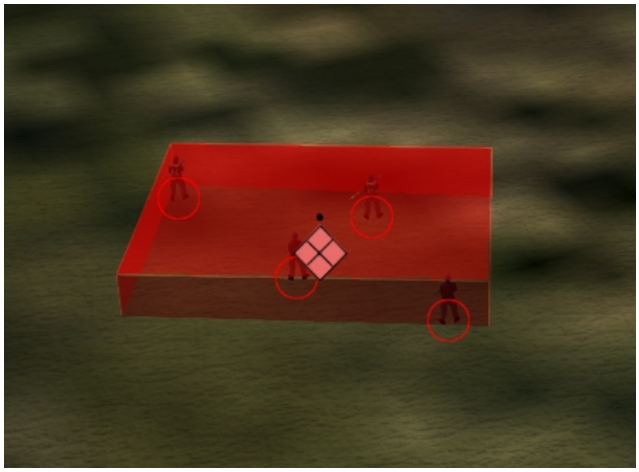
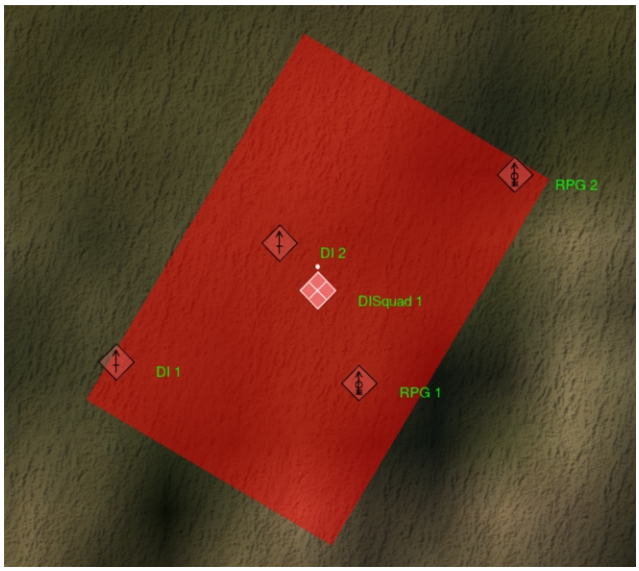


Figure 171 shows a unit bounding volume in Plan View observer mode, with ghosting enabled.

Figure 171. 2D bounding volume



You can display unit icons without showing bounding volumes and vice versa.

➤ **To configure the display of unit icons and bounding volumes:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 168 on page 447).
3. To enable the display of unit icons, bounding volumes, or both, select Show Units.
4. To display bounding volumes for collapsed units, select Show Bounding Volumes.
5. To restrict the display of bounding volumes to the selected units, select Show Bounding Volumes Only for Selected Units.
6. To display unit icons in 3D observer modes, select Show Aggregate Icon (3D Only).

> You can also enable and disable units by choosing **Settings > Units**.

19.4.1 Specifying the Color Scheme for Unit Bounding Volumes

Unit bounding volumes can be displayed using primary colors or MIL-STD colors.

➤ **To specify the color scheme to use for unit bounding volumes:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 168 on page 447).
3. Select the Force Coloring Settings option that you want (Use Primary Colors or Use MIL-STD Icon Colors).



20. Selecting Objects

This chapter describes ways to select simulation objects and tactical graphics.

20.1. The Objects List Panel 452
20.2. Selecting Simulation Objects, Tactical Graphics, and Props 453
- 20.2.1 Selecting Objects in the Window 455
- 20.2.2 Selecting Objects in the Objects List Panel 456
- 20.2.3 Selecting a Simulation Object in the Object Console Summary Panel 456
- 20.2.4 Selecting the Parent Unit of an Entity 457
- 20.2.5 Unselecting Objects 457

20.3. Using Selection Groups 457
- 20.3.1 Creating a Selection Group 458
- 20.3.2 Selecting a Selection Group 459
- 20.3.3 Adding Objects to a Selection Group 460
- 20.3.4 Removing Objects from Selection Groups 460
- 20.3.5 Renaming a Selection Group 460
- 20.3.6 Deleting a Selection Group 460

20.1. The Objects List Panel

There is a great deal of similarity in how you select, move, and view the different types of objects. You can manage them on the terrain, to some extent from the keyboard, and in the Objects List Panel, which is the central repository of object information.

The views in the Objects List Panel are:

- Object View: displays simulation objects in the simulation in a hierarchical list by force. Also lists tactical graphics (Figure 172).
- Embarkation View: displays the embarkation relationships of simulation objects (Figure 172).
- Overlays View: lists overlays and the objects that are on them (Figure 173).
- Selection Groups View: lists selection groups and their members (Figure 173).

Figure 172. Objects List Panel – Objects List View and Embarkation View

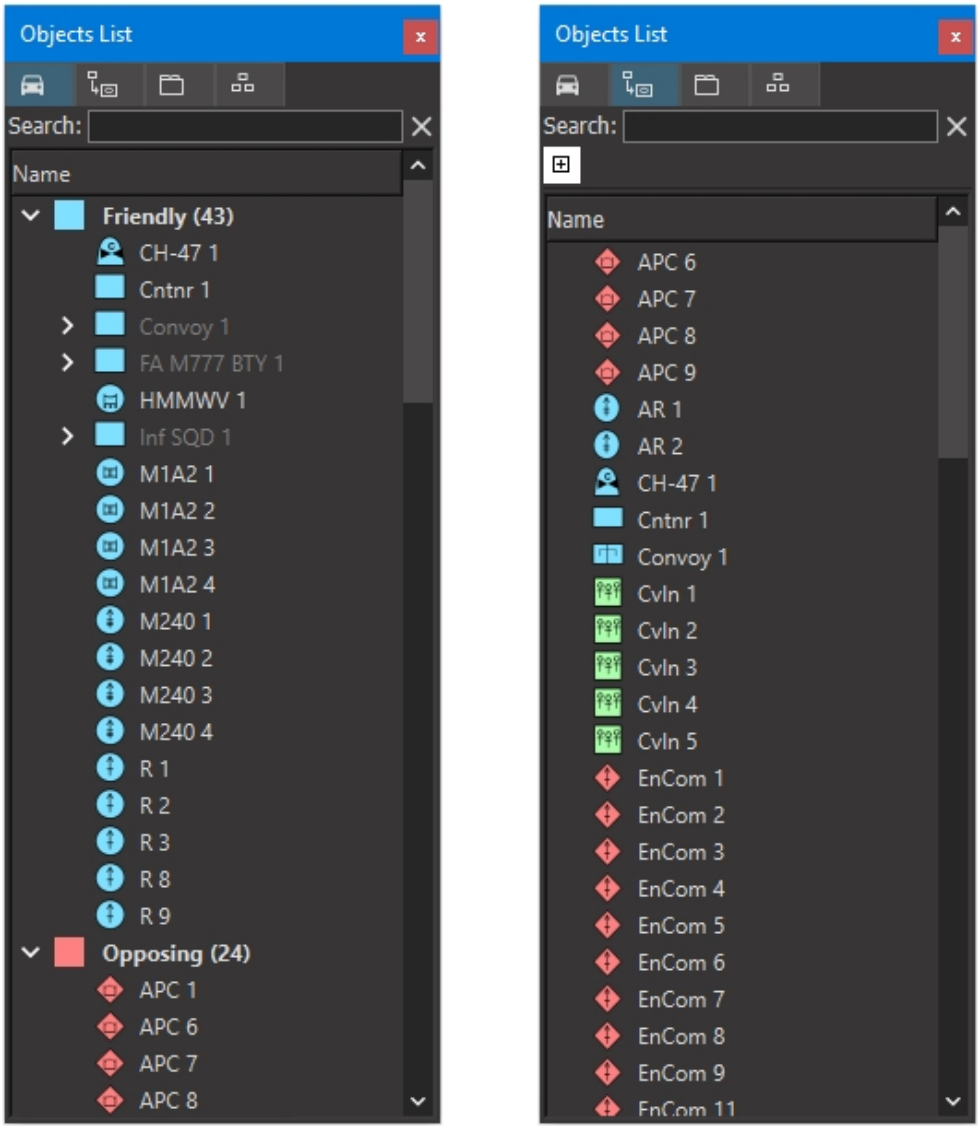
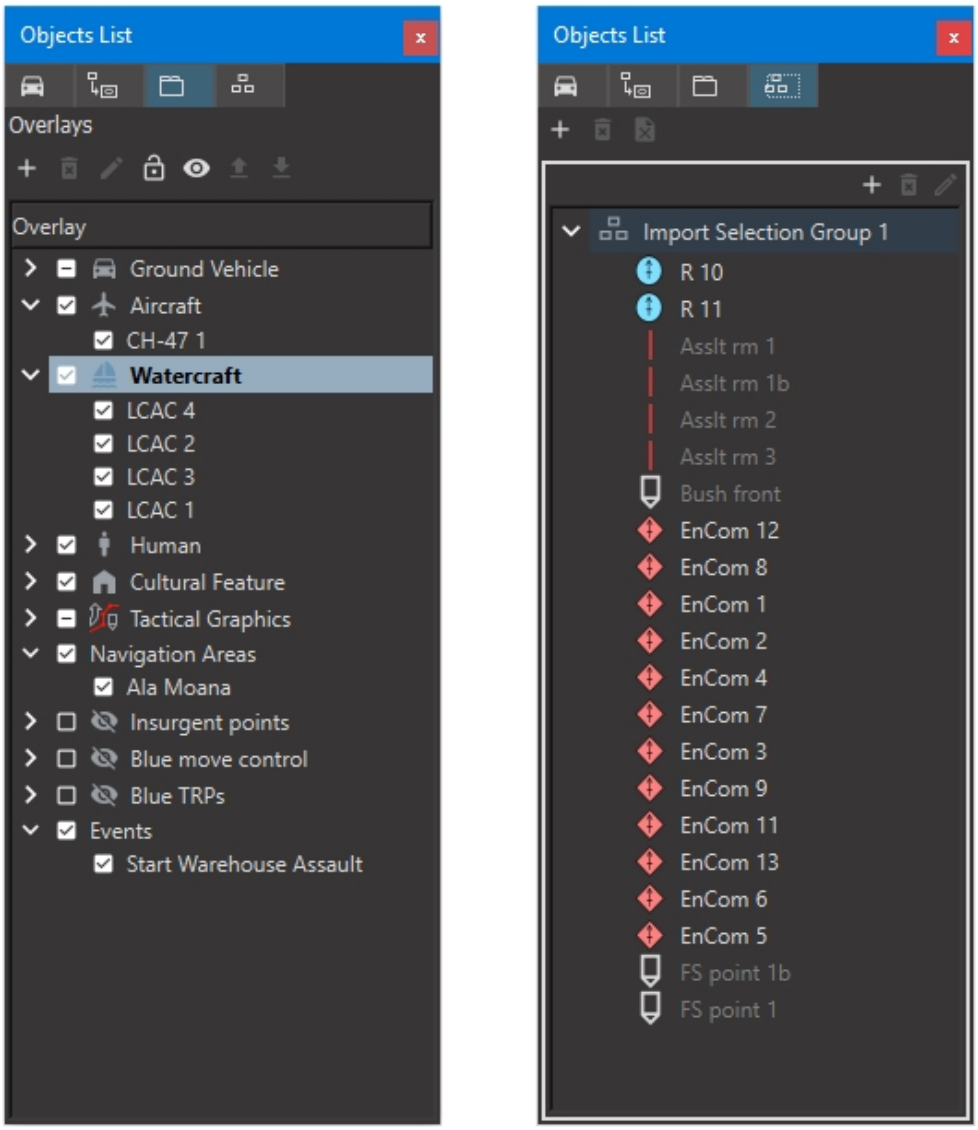


Figure 173. Objects List Panel – Overlays View and Selection Groups View



The objects in the objects list panel are identified by their 2D icon and their name. You can use images other than the default icons to identify objects. For details, see "Using an Alternative Icon List" on page 1344.

20.2. Selecting Simulation Objects, Tactical Graphics, and Props

VR-Forces displays the following kinds of objects that you might want to select:

- Simulation Objects. Any of the entity and unit types that you can create.
- Cultural features. Objects that have the Cultural Feature tag on the Create Object Palette.
- Tactical graphics. Control objects and the points, lines, areas, text, and symbols on the Create Object Palette and Create menu.

- Props. Any of the props that you can create from the Props Palette or which are already part of the terrain.

You can select all types of objects in the window. You can select simulation objects, cultural features, and tactical graphics in the Objects List Panel (Figure 174 below). You can select simulation objects in the Object Console Summary Panel, and you can select props on the Props page of the Terrain Settings dialog box (Figure 175).

Figure 174. Selected 3D model

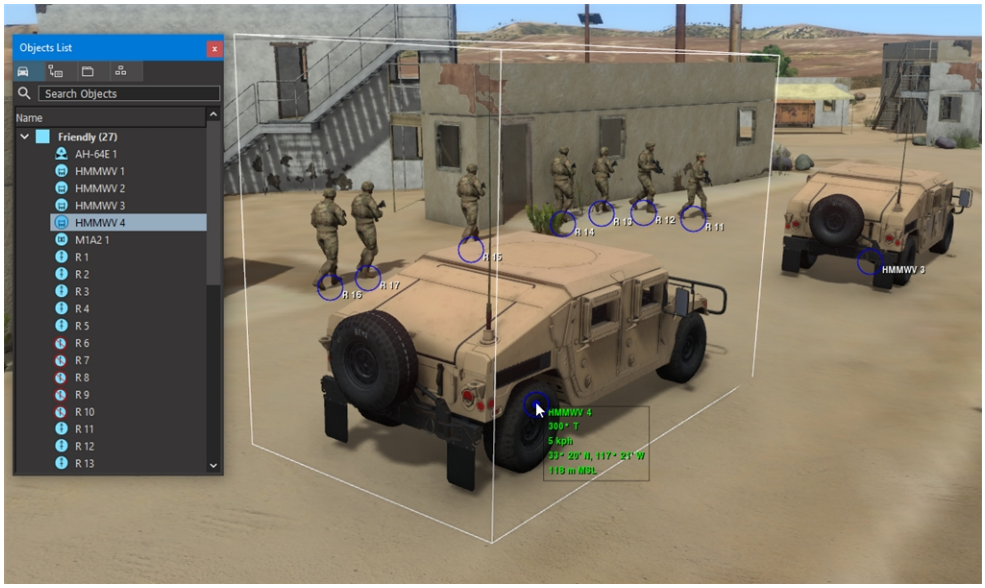
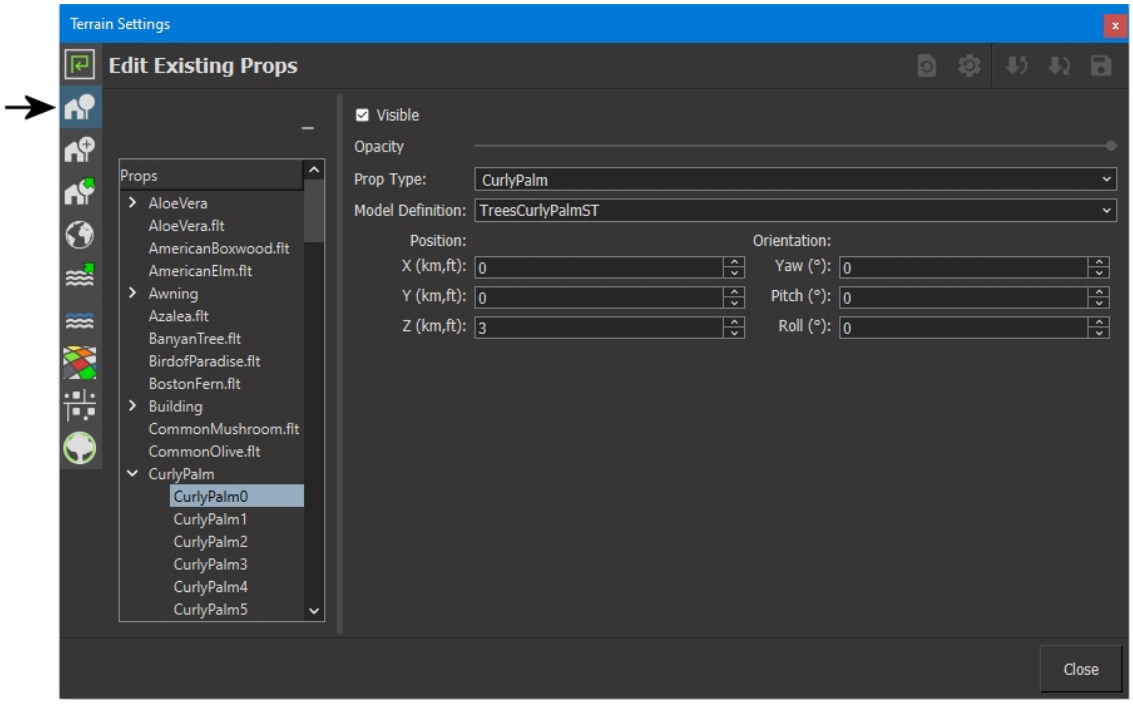
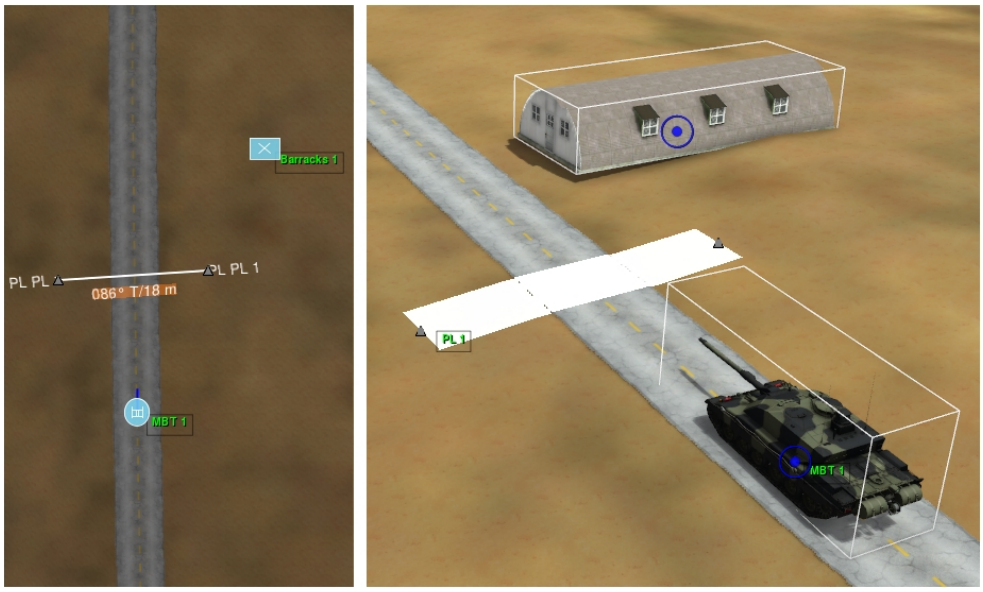


Figure 175. Terrain Settings dialog box, Props page



When you select a simulation object or tactical graphic in the 2D view, its border color changes. In the 3D views, all selected 3D objects display a bounding box (Figure 176). Tactical graphics outlines are highlighted.

Figure 176. Selected objects (2D and 3D)



20.2.1 Selecting Objects in the Window

To select objects in the window, use any of the following mouse actions:

Mouse Action	Result
Click the object.	Selects an object (and unselects any other selected objects).
Ctrl+click the object.	Selects an object while keeping existing selections.
Double-click the object.	Selects the object and makes it editable. This is the equivalent of selecting an object and choosing **Objects > Edit**.
Alt+drag a bounding box.	Selects multiple objects by dragging a selection box around them.
Right-click the object.	If no objects are selected, right-clicking selects an object and displays a menu. If one or more objects are selected, right-clicking a selected object maintains the selection and displays a menu. Right-clicking an unselected object changes the selection to the object you right-clicked.

When you select an object in the window, the views on the Objects List Panel add a highlight to the entry for the object to indicate that it is selected. Note that when you select multiple objects with a selection box, the Objects List does not highlight the selected items until you release the mouse button.

When visual item are displayed for simulation objects (radar coverage, range rings, sensor contacts, task visualizations, track histories, or task visualizations) and you then select multiple objects, the visual item displays only for the first selected object. To display them for more than one object at a time, use the appropriate pin option in the Objects > Decorations submenu.

20.2.2 Selecting Objects in the Objects List Panel

➤ **To select an object in the Objects List Panel:**

1. Select a view in the Objects List Panel (Figure 172 on page 452).
2. Locate the object you want to select. You can filter the list to make it easier to find particular objects. (For details, see "Filtering the Object Lists" below.) Expand units if necessary. (For details, see "19.2. Expanding and Collapsing Units" on page 444.)
3. Click the entry for the object. The selected object is highlighted in the Objects List Panel and on the terrain.
4. Optionally, select multiple objects:
   - **Shift**+click to select a contiguous range of simulation objects.
   - **Ctrl**+click to select multiple non-contiguous simulation objects.

Filtering the Object Lists

You can filter the display in the Objects List.

➤ To filter the display of objects, type a string in the Search box at the top of the list (Figure 172 on page 452).

20.2.3 Selecting a Simulation Object in the Object Console Summary Panel

The Object Console Summary Panel lists messages that have been sent to simulation objects from the simulation engine, from a simulation object's plan, from other simulation objects, and from scripts. (For information about the Object Console Summary Panel, see "21.9. Configuring Object Console Messages" on page 483.)

➤ **To select a simulation object in the Object Console Summary Panel:**

1. Right-click a message from the simulation object you want to select.
2. On the shortcut menu, choose Select Entity.

20.2.4 Selecting the Parent Unit of an Entity

You can easily locate the parent unit of a subordinate entity.

➤ To select an entity's parent unit, right-click the entity on the terrain or in the Objects List and then choose Select Superior. If the object is the subordinate of a unit, the parent unit is selected in the Objects List.

20.2.5 Unselecting Objects

To unselect objects in the window, choose one of the options in Table 24.

Table 24. Unselecting objects on the terrain

Action	Result
Click in the window away from the selected object or objects.	Unselects all objects.
Right-click.	Exits edit mode, but object remains selected.
Click a selected object.	If multiple objects were selected, unselects all except the object you clicked.
Ctrl+click a selected object.	Unselects the object. If multiple objects were selected, does not unselect other objects.

To unselect objects on the Objects List Panel, choose one of the options in Table 25.

Table 25. Unselecting objects in the Objects List Panel

Action	Result
If one object is selected:	
Ctrl+click the object.	Unselects that object.
If a group of objects is selected:	
Ctrl+click an object.	Unselects the object without affecting other selections.
Click an object.	Selects the clicked object and unselects all other objects.

20.3. Using Selection Groups

Selection groups are named subsets of the objects in an exercise. Their purpose is to allow you to quickly select specific objects without needing to locate them on the map or the Objects List View in the Objects List Panel and select them individually. Using selection groups can be helpful if you want to quickly select a group of objects that do not fit neatly

into the default filter options on the Objects List Panel views, for example, a mix of friendly and opposing forces and tactical graphics. The group definitions are saved and loaded as part of the scenario.

An object can be in multiple selection groups.

20.3.1 Creating a Selection Group

You can create selection groups in any of the following ways:

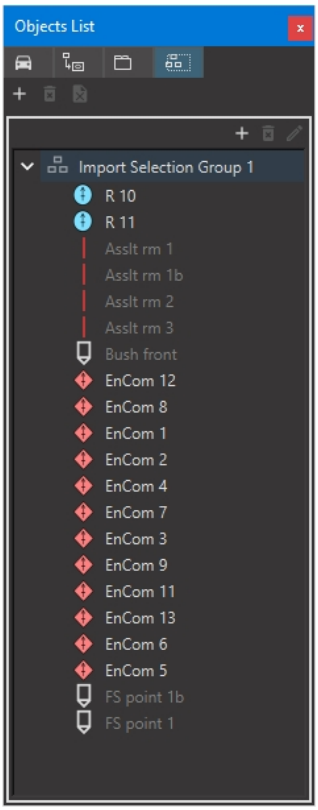
- In the Selection Groups View on the Objects List Panel.
- From Selection Groups menu options.
- Using keyboard shortcuts.

When you add a simulation object group to a scenario, VR-Forces automatically creates a selection group for all the objects in the simulation object group. For information about simulation object groups, see "14.14. Simulation Object Groups" on page 400.

➤ **To create a selection group in the Selection Groups Panel:**

1. On the Objects List Panel, select the Selections Groups View (Figure 177).

Figure 177. Selection Groups tab



2. Click the Add button (+). The New Selection Group dialog box opens.

3. Type a name for the selection group.
4. Click OK. The group is added to the group list.

Creating a Selection Group from a Selection Group Menu Command

When you create a selection group from a Selection Group menu command, you can combine the procedures for creating a new selection group and adding an object to it.

➤ **To create a selection group from a Selection Group menu command:**

1. Select the objects you want to put in a new selection group.
2. Choose **Objects > Selection Groups > Add to New Selection Group**. The New Selection Group dialog box opens.
3. Type a name for the selection group.
4. Click OK. The selection group is created and the selected objects are added to it.

Creating a Selection Group from a Keyboard Shortcut

You can quickly create a selection group with a default name using a keyboard shortcut. When you create a selection group from the keyboard:

- If no objects are selected, an empty selection group is created.
- If objects are selected, a selection group is created with these objects as members.
- If the keyboard shortcut is for a selection group that already exists, a new selection group replaces the previous one. If the previous group had members and the new one is created without any objects selected, the new group is empty.

➤ To create a selection group from the keyboard, press **Ctrl** + *number*, where *number* is 1, 2, 3, 4, 5, 6, 7, 8, 9, or 0. A selection group is created named Selection Group *number* (0 creates Selection Group 10). (The keypad numbers do not support this operation.)

20.3.2 Selecting a Selection Group

When you select the objects in a selection group, they are selected in the views on the Objects List Panel and in the display window.

➤ **To select a selection group, use one of the following methods:**

- In the Selection Groups view, select a selection group in the Group list.
- Choose **Objects > Selection Groups > Select Selection Group >** *group_name*, where *group_name* is one of the groups listed on the pull right menu.
- If selection groups named with the format Selection Group *number* exist, pressing a number key selects the corresponding selection group. (The keypad numbers do not support this operation.)

20.3.3 Adding Objects to a Selection Group

➤ **To add objects to a selection group:**

1. Select the objects you want to add to a selection group.
2. Choose **Objects > Selection Groups > Add To Selection Group >** *group_name*, where *group_name* is one of the groups listed on the pull right menu.

20.3.4 Removing Objects from Selection Groups

There are two ways to remove an object from a selection group. If you remove all the objects from a selection group, the group continues to exist.

➤ **To remove an object from a selection group:**

1. Select the objects you want to remove from a selection group.
2. Choose **Objects > Selection Groups > Remove from Selection Group >** *group_name*, where *group_name* is one of the groups listed on the pull right menu.

➤ **To remove an object from a selection group in the Selection Groups view:**

1. Select the selection group from which you want to remove an object.
2. Select the object you want to remove.
3. Click the Remove Selected from Selection Group button ().

20.3.5 Renaming a Selection Group

You can rename selection groups. If you give a group the name Selection Group *number*, where *number* is 1, 2, 3, 4, 5, 6, 7, 8, 9, or 0, then when you press that number then the corresponding selection group becomes selected. (The keypad numbers do not support this operation.)

➤ **To rename a selection group:**

1. In the Selections Group view, double-click the name of the group you want to rename. It becomes editable.
2. Type a new name for the selection group.
3. Press **Enter**.

20.3.6 Deleting a Selection Group

Deleting a selection group does not delete the objects from the scenario. However, if you delete all the members of a selection group, the selection group is also deleted.

➤ **To delete a selection group:**

1. In the Selections Group view, select the group you want to remove.
2. Click the Delete button ().



21. Viewing Information About Objects

This chapter describes how to get and display information about simulation objects.

21.1. Displaying Information About an Object 462
21.1.1 Information Dialog Boxes for Entity-Level Scenarios 462
21.1.2 Information Dialog Boxes for Aggregate-Level Scenarios 465
21.1.3 Information Dialog Boxes for Tactical Graphics 467
21.1.4 Viewing Information for Multiple Objects 468
21.2. Displaying Entity Labels 469
21.2.1 Turning Entity Labels On and Off 471
21.2.2 Pinning Symbol Decorations to the Window 472
21.2.3 Symbol Decorations 473
21.2.4 Configuring Shadowing and Background Color for 2D Labels 476
21.3. Using Extended Labels 477
21.3.1 Adding an Extended Label 478
21.3.2 Changing an Extended Label's Index 478
21.3.3 Deleting an Extended Label 479
21.4. Customizing 2D Icon Colors 479
21.5. Changing Force Colors 479
21.6. Displaying HAT Lines for Entities (3D Only) 480
21.7. Displaying Entity Resources 481
21.7.1 Viewing Numerical Resource Data 481
21.8. Displaying Bounding Volumes 481
21.8.1 Enabling and Disabling Bounding Volumes for Entities 482
21.8.2 Showing Bounding Volumes Only for Selected Entities 482
21.9. Configuring Object Console Messages 483
21.9.1 Setting the Notification Level for Console Messages 483
21.9.2 Viewing All Console Messages 484
21.9.3 Answering Questions from Scripted Tasks 485
21.9.4 Saving Console Messages 485

21.9.5 Displaying an Object Console Warning Icon 486
21.9.6 Clearing the Object Console 487
21.10. Viewing Object Counts 487

21.1. Displaying Information About an Object

You can display information about individual objects, including entities, units, environmental objects, and Stealths. You cannot edit this information. You can also display entity labels. For details about entity labels, see "21.2. Displaying Entity Labels" on page 469.

➤ **To display an object information dialog box:**

1. Select an object.
2. Choose **Objects > Information**, or press **I**.

Other methods for opening the Information dialog box are:

- Right-click a message in the Objects Console Summary Panel and choose **Information** from the context menu, or double-click the line.
- Click the Show Information button (ⓘ) on the Last Selected Object Panel or Command Panel to open the State Data page.
- Click the current task on the Last Selected Object Panel to open the Information dialog box to the Tasks page.

You can open copy the text of an item in the Value column by right-clicking on it and then choosing Copy Value.

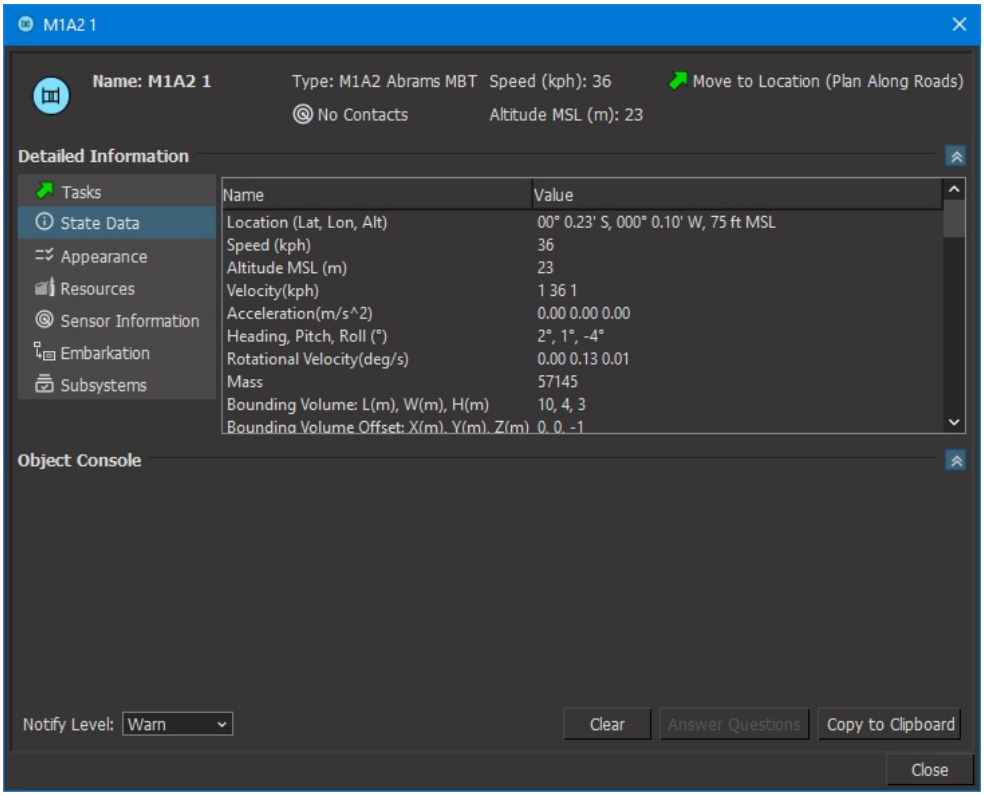
21.1.1 Information Dialog Boxes for Entity-Level Scenarios

An information dialog box in an entity-level scenario (Figure 178) has the sections:

- Summary:
  - Simulation object icon
  - Simulation object name
  - Current task
  - Sensor information
  - TAS (true airspeed)
  - Altitude
- Detailed information
- Object console

You can expand and contract the detailed information and object console sections to manage the amount of screen space used by the dialog box.

Figure 178. Information dialog box for simulation object in entity-level scenario



The Detailed Information section for simulation objects in an entity-level scenario has these pages:

- Tasks. Shows the currently assigned task, subtasks that these tasks have initiated, and any enabled reactive tasks. You can cancel a task that the entity is currently performing by clicking the Cancel icon. You can skip other non-reactive tasks using the context menu. If a task has a graphical status output, there is an additional tab for it in the Tasks page and you can see a graphical display. All scripts that use the FSM packages or the Behavior Engine have graphical status output.
- State Data. Shows location, movement values such as speed and heading, identifying information, and other status information.
- Appearance. Shows the status of appearance attributes, such as lights, firepower, and concealment.
- Resources. Shows that amount of fuel and ammunition available.
- Sensor Information. Shows information about simulation objects being detected.
- Emitters. Shows details about emitters, if applicable.
- Acoustics. Shows details about active sonar and propulsion noise, if applicable.
- Embarkation. Lists embarked objects, if any.
- IFF. Shows IFF information, if applicable.

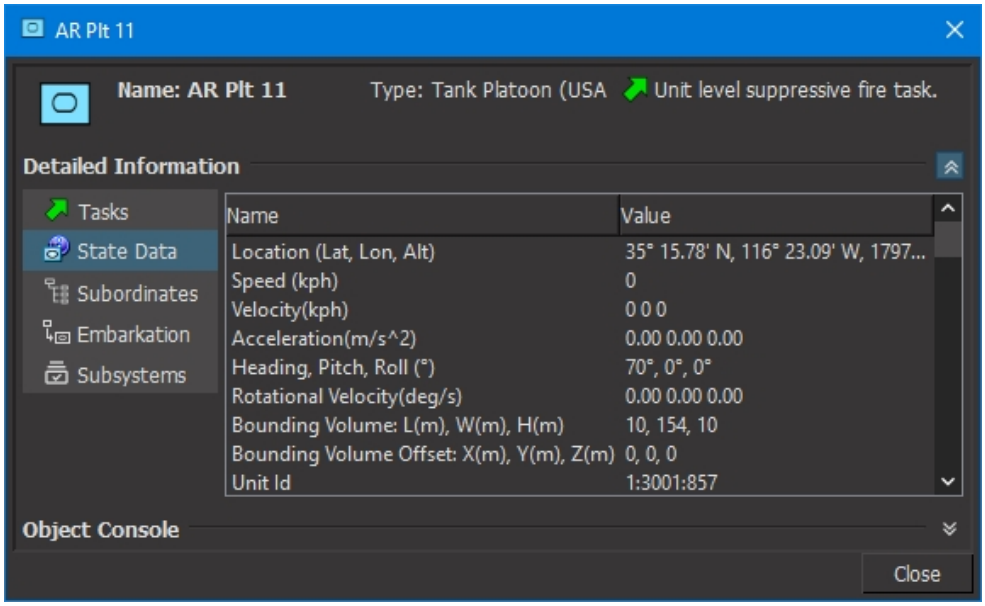
- Subsystems. Lists the entity's subsystems and indicates whether they are enabled, if applicable.

> The pages listed for Detailed Information are context sensitive. If a page does not apply to an entity type, it is not shown.

The Detailed Information section for units in entity-level scenarios (Figure 179) has the pages:

- Tasks.
- State Data.
- IFF (if applicable).
- Subordinates. Lists the subordinates.
- Acoustics (if applicable).
- Embarkation.

Figure 179. Information dialog box for units in entity-level scenarios



Airspeed for Rotary- and Fixed-Wing Aircraft

The State Data page for rotary-wing and fixed-wing aircraft displays this speed information:

- Speed. Ground speed is the velocity of the aircraft.
- True airspeed. Calculated based on ground speed, temperature, pressure, and altitude.
- Indicated airspeed. Indicated airspeed (IAS) is calculated as 2% less than true airspeed for every 300 ft of altitude.
- Mach.

If you want to substitute your own calculations for airspeed, you can do so using the VR-Forces API.

Indicated airspeed, true airspeed, and mach are user-created state properties that are defined in *./data/simulationModelSets/EntityLevel/vrfSim/platforms/Fixed_Wing_Aircraft.ope* and *./data/simulationModelSets/EntityLevel/vrfSim/platforms/Rotary_Wing_Aircraft.ope*.

```
(DtRwReal IndicatedAirspeed 0.0 publish)
  (DtRwReal TrueAirspeed 0.0 publish)
  (DtRwReal MachNumber 0.0 publish)
```

Therefore, you can change the default value.

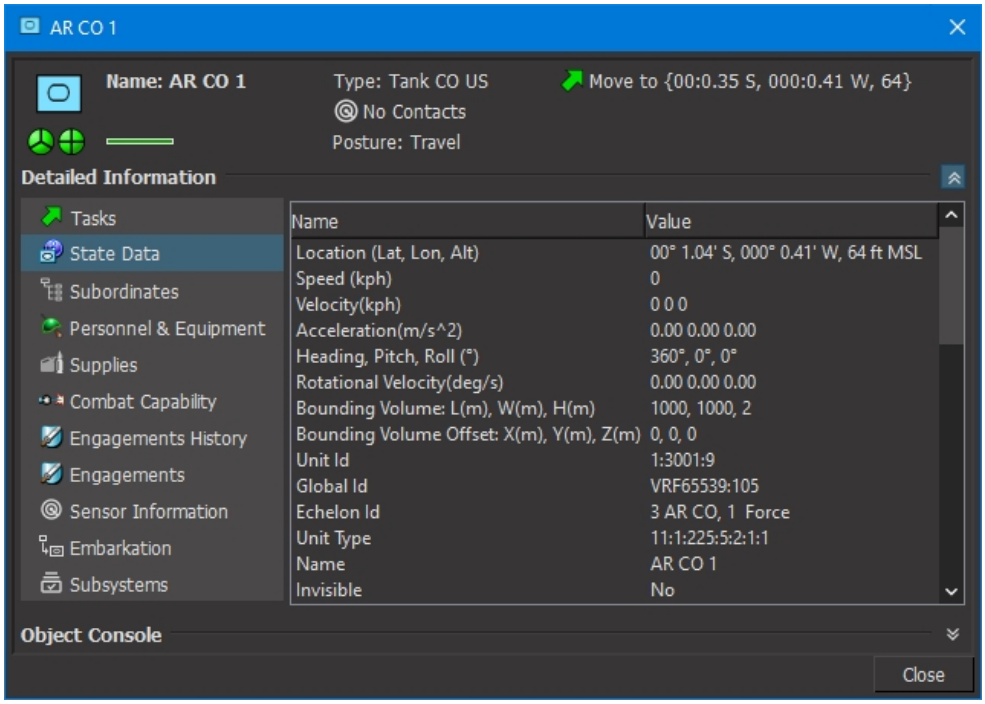
You can change the unit of measurement for true airspeed and indicated airspeed on the Display Settings dialog box, Display Unit Settings page.

21.1.2 Information Dialog Boxes for Aggregate-Level Scenarios

An information dialog box for units in an aggregate-level scenario (Figure 180) has the sections:

- Summary:
  - Simulation object icon
  - Simulation object name
  - Current task
  - Sensor information
  - Posture
  - Health status
  - Health and weapon status gumballs
- Detailed information
- Object console

Figure 180. Information dialog box for simulation object in aggregate-level scenario



The summary section for units in aggregate-level scenarios (Figure 180) includes a health status bar and gumball icons that display information about the health of the simulation object. Simulation object icons can also display gumballs. (Gumballs are disabled by default. Health status is enabled.)

Figure 181 describes the status represented by different gumball states.

Figure 181. Gumball charts



Figure 182 illustrates the health status bar and gumball charts for a simulation object icon.

Figure 182. Icon with gumballs



You can expand and contract the detailed information and object console sections to manage the amount of screen space used by the dialog box.

The Detailed Information section for simulation objects in aggregate-level scenarios has the pages:

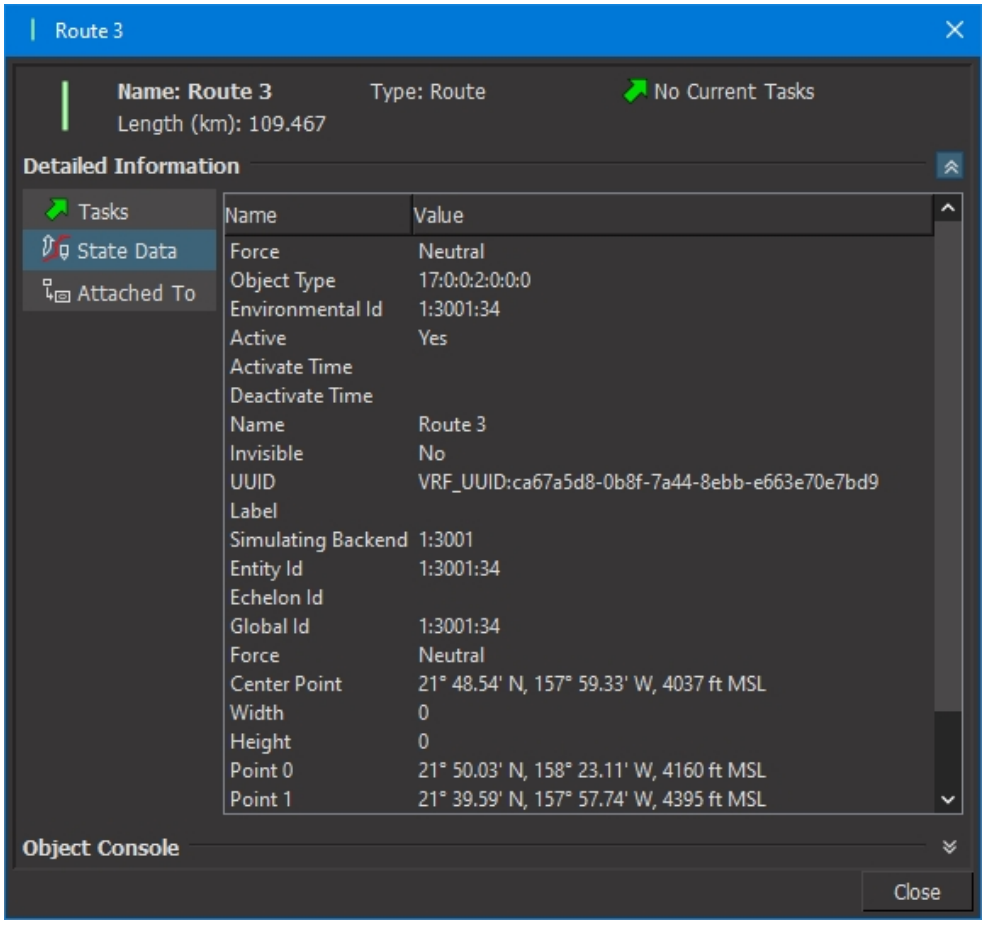
- Tasks.
- State Data.
- IFF.
- Subordinates. Lists the subordinates.
- Personnel and Equipment. The types of personnel and number of each. Casualties, by category.
- Supplies. Food, water, fuel, and ammunition. The Supplies tab displays information when a simulation object is receiving supplies or providing them to another simulation object. When a simulation object is receiving supplies, an up arrow is displayed next to the supply it is receiving.
- Combat Capability. Combat power by category (anti-air, anti-personnel, and so on) and defensive capability, by category.
- Engagements. Current attack and defense status.
- Sensor Information. The Sensors tab shows information about simulation objects being detected. The Jamming tab shows if the simulation object is being jammed (electronic warfare) or is jamming other simulation objects.
- Emitters.
- Acoustics.
- Embarkation.

21.1.3 Information Dialog Boxes for Tactical Graphics

The detailed information section for an environmental object (Figure 183), such as a route, has these pages:

- State Data. Shows the force and coordinates.
- Attached To. The simulation object this object is attached to, if any.

Figure 183. Environmental object information dialog box



21.1.4 Viewing Information for Multiple Objects

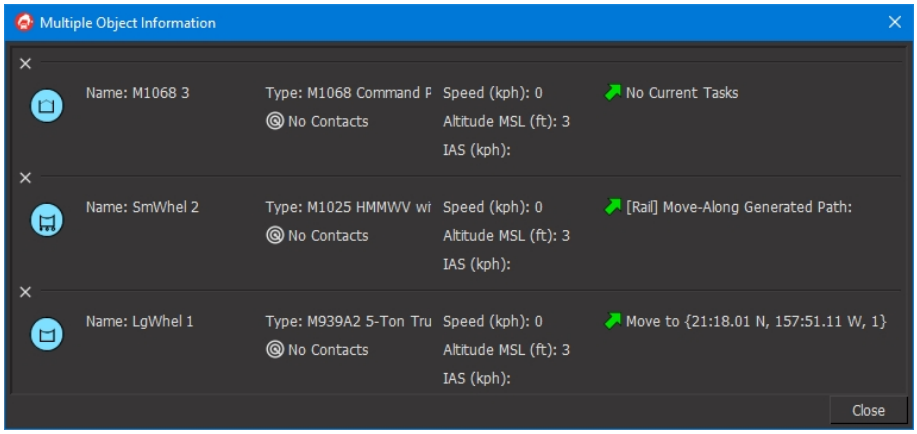
You can select multiple objects and view summary information for all of them. From the summary information window, you can jump directly to an object and you can open its Information dialog box.

If you select more than 20 simulation objects, the Multiple Object Information window displays information about the first twenty and then there is a button at the bottom of the window that allows you to load more objects. You can also close the information for each select object, which automatically loads another object's information.

➤ **To view information about multiple objects:**

1. Select the objects.
2. Choose **Objects > Information**, or press **I**. The Multiple Object Information window opens (Figure 184).

Figure 184. Multiple Object Information window



3. To zoom to the extents of an object, click its icon.
4. To open an Information dialog box for an object, click any of the details about the object, such as its name.
5. If you selected more than 20 objects and you want to see more of them in the Multiple Object Information window, click Show Next *x* of *y*.
6. To remove an object from the window, click the ✕ button.

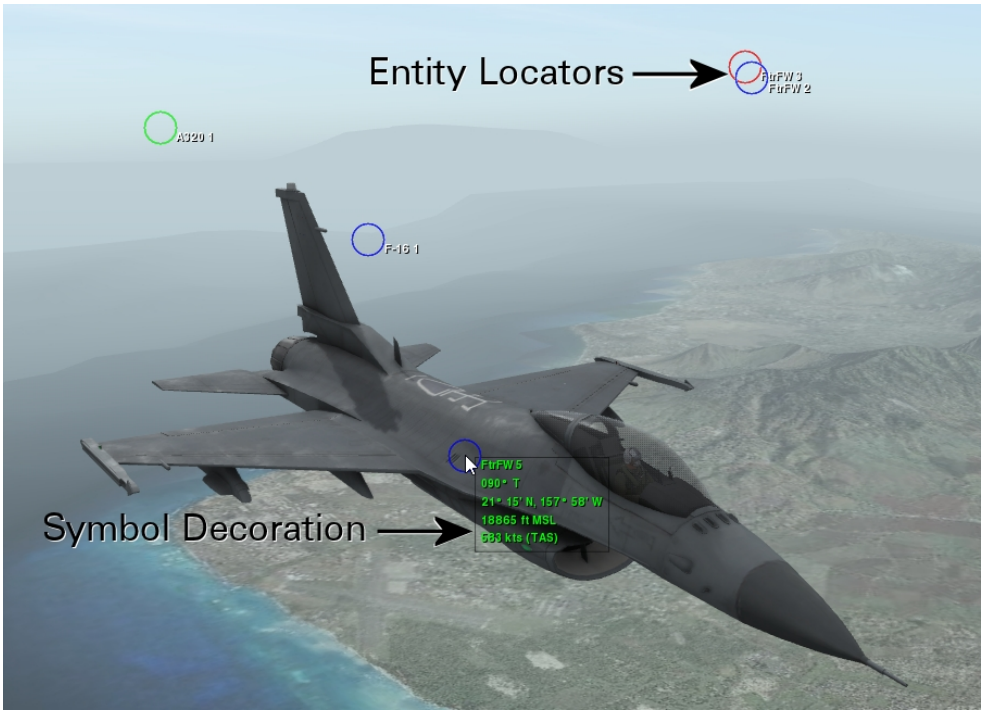
21.2. Displaying Entity Labels

VR-Forces provides extensive information about simulation objects in the Information dialog box ("21.1. Displaying Information About an Object" on page 462). It also provides an abbreviated set of information in the form of simulation object labels, also known as symbol decorations.

The symbol decorations that VR-Forces displays for simulation objects help you locate and identify entities (Figure 185). For 3D models, the decoration consists of a locator circle, which is always visible when labels are enabled, and a text label that you can display either all of the time or only when you mouse over an entity. All entities within the view have locator circles, regardless of whether the entity is clearly visible in the window or too far away to see. The color of an entity's label represents its force — the default colors are blue for friendly, red for opposing, and green for neutral.

> You can configure the attributes of the entity label, such as font size, and the height and width of the background by editing the entityInfo model definition in the Visual Models Editor. For details about editing model definitions, see "Creating and Editing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.

Figure 185. Symbol decorations for 3D models



Symbol decorations are a set of individually enabled text labels (Figure 186). Entity labels for 2D icons are supported both in the 2D view and the 3D view. If you resize 2D icons, the labels are also resized.

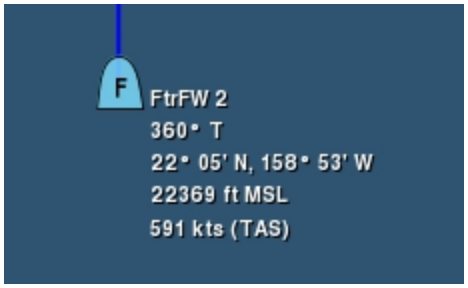
Symbol decorations can display the simulation object's:

- Acceleration
- Altitude above ground level – This displays only for aircraft, space, and lifeform entities.
- Altitude above mean sea level – This displays only for aircraft, space, ground vehicle, subsurface, and lifeform entities (not surface watercraft).
- Contact Radius and Contact Radius Indicator
- Heading and Heading indicator
- Label – If extended labels exist, each label can be displayed separately.
- Location
- Name
- Observed ... – Information observed about the entity.
- Orientation
- Reported By
- Speed – For aircraft, you can also show the ground speed (GS), indicated airspeed (IAS), true airspeed (TAS), and Mach number (MACH).
- TDL ... – Tactical data link information.

- Time of Last Contact
- Time Since Last Report
- Velocity indicator
- Weather – Elevation configuration, cloud state, precipitation intensity, visibility, wind direction, and wind speed.

If spot reports are enabled, you can display labels for spot reports. For details, see "9.1.6 Displaying Labels for Spot Reports" on page 296.

Figure 186. Label for a 2D icon



You can specify the order of symbol decorations. See exampleSymbolDecoration in the *VR-Forces Developers Guide* to learn how to do this.

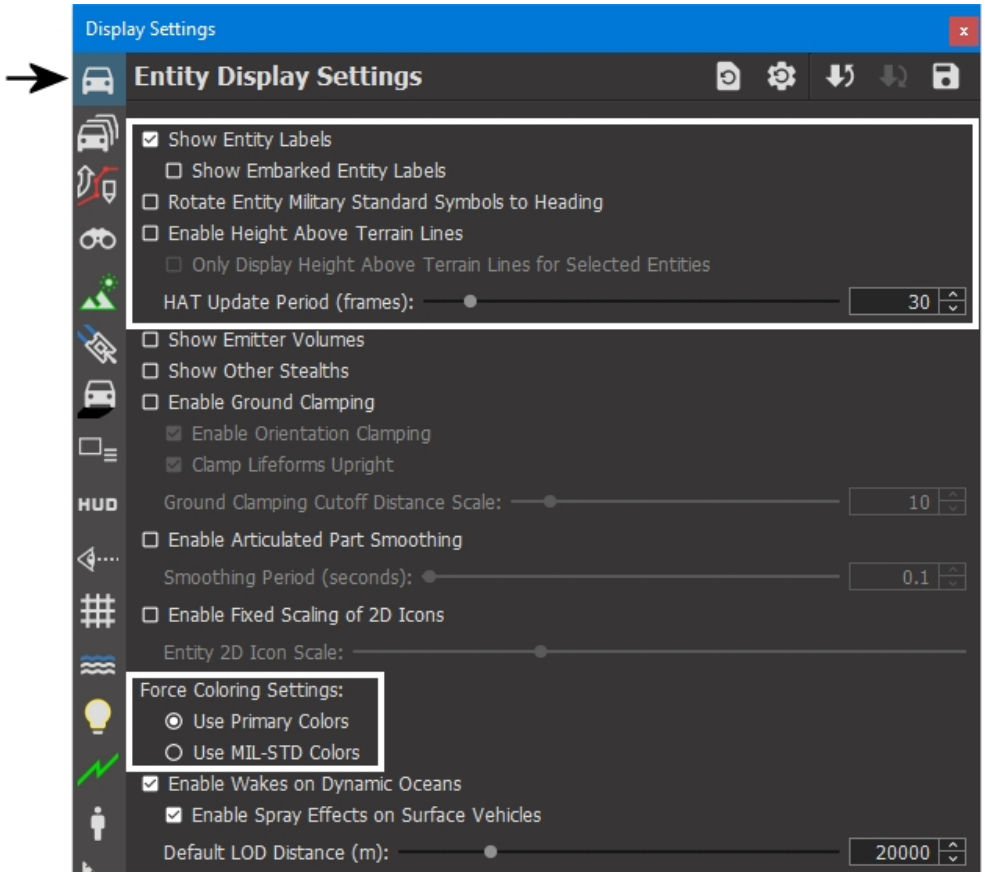
21.2.1 Turning Entity Labels On and Off

➤ **To turn the display of entity labels on or off:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 187).
3. Select or clear the Show Entity Labels check box.

You can also enable and disable entity labels by choosing **Settings > Entity Labels**, pressing **L**, or by clicking the Simulation Object Labels button ( ) on the Display Settings Toolbar.

Figure 187. Display Settings, Entity Display Settings page



Turning Embarked Entity Labels On and Off

You can choose whether to display the labels for embarked entities.

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 187 above).
3. Select or clear the Show Embarked Entity Labels check box.

> You can also enable and disable entity labels by choosing **Settings > Embarked Entity Labels** or by clicking the Embarked Object Labels button ( ) on the Display Settings Toolbar.

21.2.2 Pinning Symbol Decorations to the Window

You can pin symbol decorations to the window, making them stay visible after you move the cursor away from the entity. The pin applies to both the 2D and 3D views. The pin icon indicates the state of each entity's label:

- (sideways). The label for each selected entity is not pinned.
- (upright). The label for each selected entity is pinned.

- (angled). Multiple entities are selected and only some of their labels are pinned.

➤ **To pin or hide a symbol decoration in the window:**

1. Right-click an entity's locator circle or the entity name in the Objects List. The context menu displays.
2. Choose **Toggle Pinned Label**.

Displaying Multiple Symbol Decorations

You can display multiple symbol decorations at one time.

➤ **To display multiple decorations, use either of the following methods:**

- Right-click and choose **Toggle Pinned Label** for each entity whose label or decoration you want to see.
- To display selected symbol decorations at all times for entities, on the Display Settings dialog box, Symbol Decoration Settings page, select the check box in the On/Off column.

21.2.3 Symbol Decorations

You can enable and disable the display of symbol decorations by label type. You can also specify which labels should show all of the time, and which should display only when you mouse over an entity. This allows you to control how much screen space is taken up by the labels. You can control the display of labels in the Display Settings dialog box.

> By default, the symbol decorations for unselected entities are hidden when the observer is a certain distance away from them. To adjust that distance, see "Adjusting Display of Symbol Decorations by Altitude" on page 475.

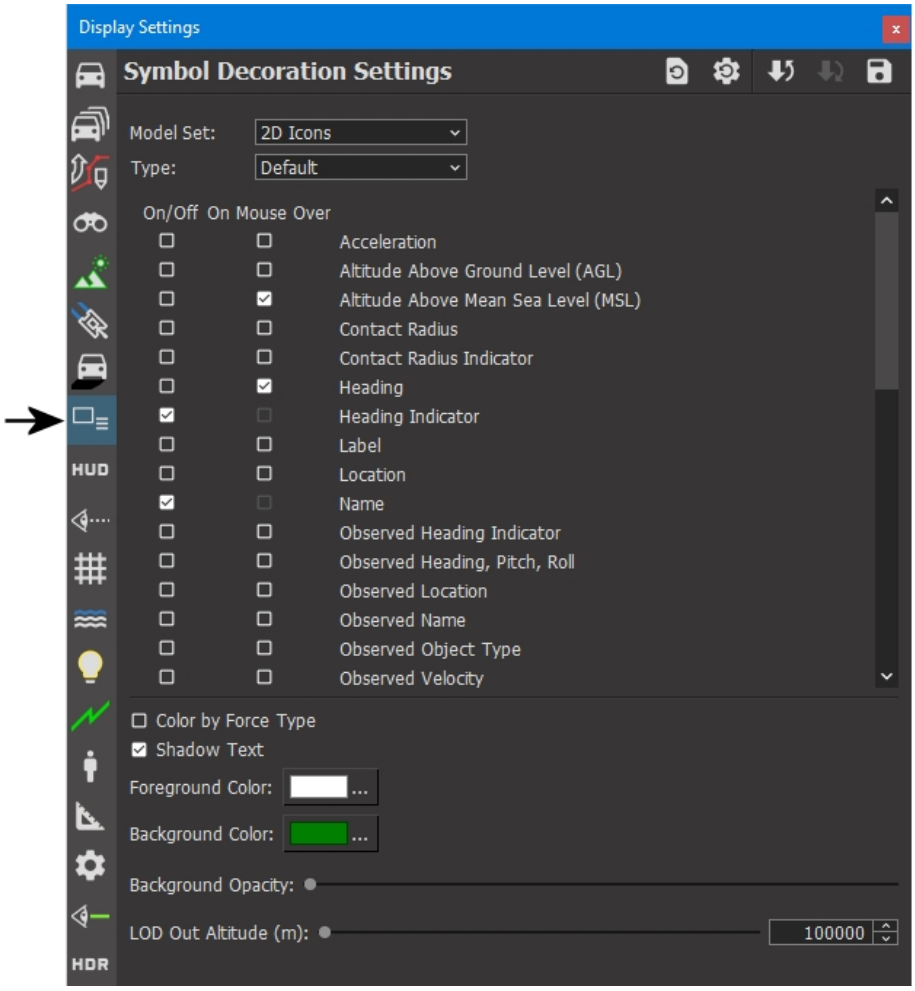
Displaying Symbol Decorations

You can control what symbol decorations display based on the model set and also whether the labels display all the time or only when you mouse over a simulation object. Note that, when you select the On/Off check box to indicate that the label should display regardless of the mouse cursor's position, the On Mouse Over check box is disabled.

➤ **To enable and disable display of simulation object labels in Display Settings dialog box:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 188).

Figure 188. Symbol Decoration Settings page



3. Select the model set where you want the labels to display.
4. Select the check box for each label that you want to display.
   - To display the label all of the time, select the check box in the On/Off column.
   - To display the label only when the cursor is over a simulation object, select the check box in the On Mouse Over column.

   Clear the check box for each label that you do not want to show.

Changing the Color for Label Text

Symbol decoration text uses the color specified by the Foreground Color setting on the Symbol Decoration Settings page. You can also specify that labels use the force color.

➤ **To change the color of labels:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 188 above).

3. To set the color to any color:
    a. Click the Foreground Color color swatch. A color picker dialog box opens.
    b. Select the color you want to use.
    c. Click OK.
4. To use the force color for labels, select the Color by Force Type check box. This setting overrides the Foreground Color setting.

Adjusting Display of Symbol Decorations by Altitude

Symbol decorations are hidden when the observer rises to a specified altitude. The default is 100,000 meters, but you can change it. If you hover the mouse over or select a simulation object, the symbol decorations display regardless of the observer's altitude. This does not affect heading or velocity indicators.

➤ **To adjust the altitude at which symbol decorations are hidden:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 188 on the previous page).
3. Use the slider or the text box to adjust the LOD Out Altitude.

> 2D icons can also be resized based on the observer's altitude. See "Scaling 2D Icons by Altitude" on page 1081.

Repositioning Labels

By default, text labels in the 2D view display below and to the right of simulation object icons. You can rotate the labels to other offset positions for a simulation object if they are cluttered together and you want to be able to read them clearly. The label can be in one of four positions: above and to the right, above and to the left, below and to the left, or below and to the right.

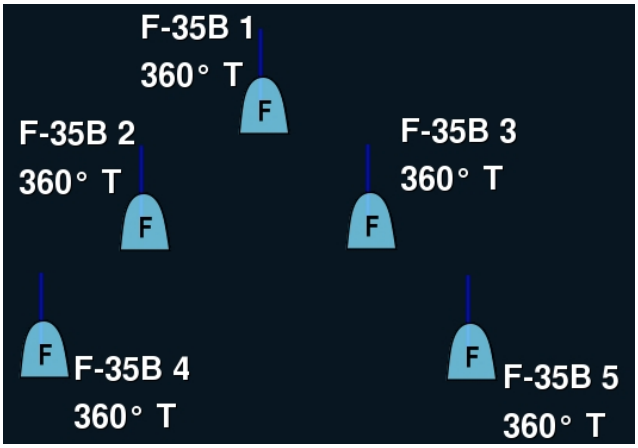
➤ To reposition a label, select the entity and then do one of the following:

- Open the context menu and then choose **Rotate Label**.
- Press **Ctrl**+**Shift**+**S**.

Repeat until the label is where you want it.

> You can select multiple entities and reposition their labels simultaneously.

Figure 189. Repositioned entity labels



21.2.4 Configuring Shadowing and Background Color for 2D Labels

Labels are drawn on a background that, by default, is transparent. However, you can increase the opacity of the background to make it visible and can change the background color.

You can add shadowing to the label text. This may improve visibility when simulation objects are on light colored terrain.

Figure 190. 2D label with background



➤ **To set the color and opacity of the label background:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 188 on page 474).
3. To change the opacity of the background, adjust the Background Opacity slider.
4. To change the color of the background:
    a. Click the Background Color color swatch. A color picker opens.
    b. Select the color you want to use for the background.
    c. Click OK.

➤ **To configure shadows for labels:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Symbol Decoration Settings page (Figure 188 on page 474).
3. Select or clear the Shadow Text check box.

21.3. Using Extended Labels

One of the ways to identify a simulation object is by its label. (For more information, see "13.2.5 Labels" on page 364.) Extended labels are additional labels that you can specify on an application-wide basis and define individually for simulation objects. The contents of each label is sent over the network as part of the simulation object's information and is displayed in its Information dialog box.

You can add and remove extended labels and change the label name. Changing a label's name does not affect the text that is associated with the label for those simulation objects that use the label.

> Extended labels are not scenario-specific. They affect all scenarios. Therefore, if you change a label's name or index, or delete it, be sure that you want this change to apply to all scenarios in which you use the extended label.

Figure 191 shows an extended label (called A Special Label) added in the Application Settings dialog box. Figure 192 shows the text for the extended label for a specific entity entered in the Edit dialog box and then displayed on the screen.

Figure 191. Extended label added to session

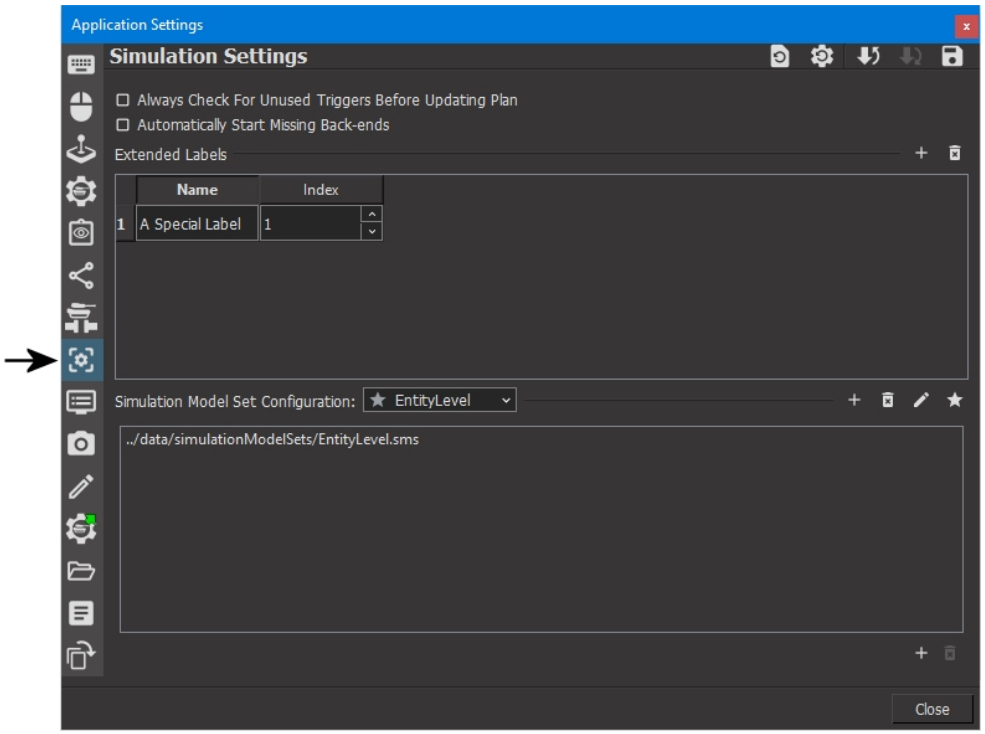
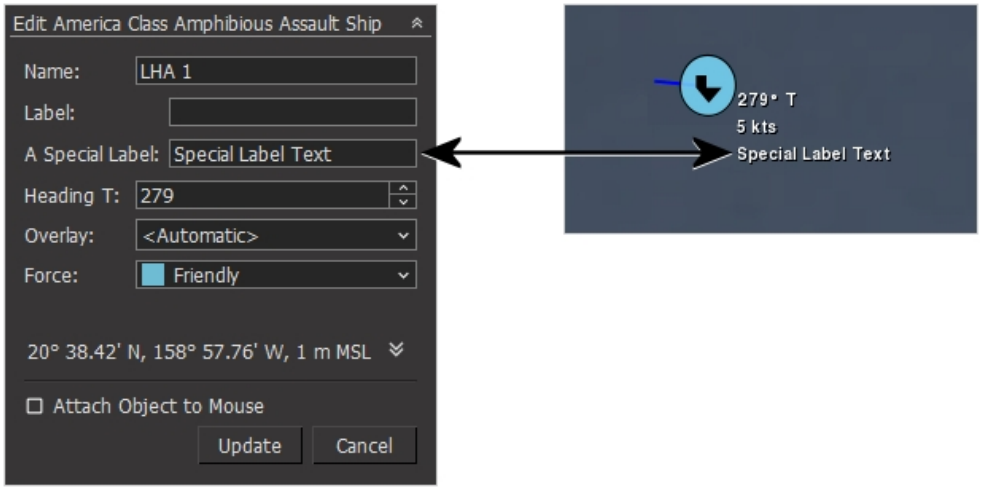


Figure 192. Extended label text for a simulation object



You can display extended labels in Plan View mode the same as other symbol decorations. For more information, see "21.2.3 Symbol Decorations" on page 473.

21.3.1 Adding an Extended Label

➤ **To add an extended label:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 191).
3. Click the Add button (+) on the Extended Labels line. A label is added to the list window. It has a generic name (for example, Label 1).
4. Double-click the label name to make it editable.
5. Type the name you want for this label.
6. Edit simulation objects to specify the text for this label.

21.3.2 Changing an Extended Label's Index

A label's index is used to store the label text and identify it when sent over the network. In Edit dialog boxes, extended labels are listed in indexed order. You can change an extended label's index. However, while changing an extended label's name does not affect the text that you save for a simulation object, changing its index could affect existing scenarios. Therefore, use caution in doing so.

➤ **To change an extended label's index:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 191 on the previous page).
3. Click the up or down arrow next to a label's index number. The index number changes and, if applicable, the label moves up or down in the list.

21.3.3 Deleting an Extended Label

Deleting an extended label affects all scenarios that use that label. (In other words, it is not a scenario-specific setting.) If you delete an extended label, it is immediately removed from all simulation objects that are using that label.

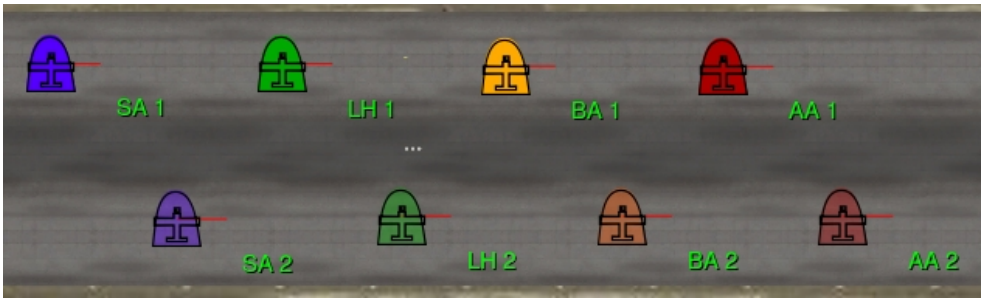
➤ **To delete an extended label:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 191 on page 477).
3. In the Extended Labels window, select the label you want to delete.
4. Click the Delete button ( ).

21.4. Customizing 2D Icon Colors

During a session, you can adjust the colors and transparency of the 2D icons. These changes only apply to the current session and VR-Forces GUI; they are not saved and they are not visible in other GUIs. You can use this to categorize the icons, for example, to differentiate the simulation objects that you are working with from those you are not.

Figure 193. 2D icons with different color and transparency settings



➤ **To change the color and transparency of a 2D icon:**

1. Select the simulation object.
2. Choose **Objects > Decorations > Set Icon Color**. The Set Icon Color dialog box opens.
3. Adjust the color and transparency (alpha channel) settings as desired.
4. Click OK.

21.5. Changing Force Colors

By default, labels, detonation lines, and other force-related simulation object graphics (including the 3D Colorized Model model set in VR-Forces) are displayed using primary colors (red, blue, and green). You can change the force coloring to use MIL-STD 2525B colors (light blue, light green, and pink (light red)).

You can also change the color of a force in the Simulation Object Editor. For details, see "68.12.2 Editing a Force" on page 1333.

➤ **To change force coloring:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. For entities and entity graphics, select the Entity Display Settings page (Figure 187 on page 472).

   For units, select the Unit Display Settings page (Figure 168 on page 447).
3. Select the Force Coloring Settings option that you want.

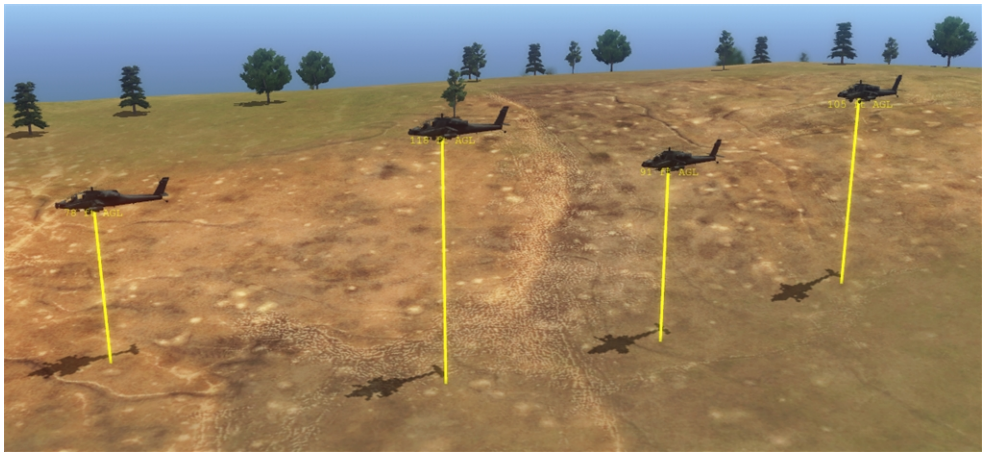
21.6. Displaying HAT Lines for Entities (3D Only)

Height-above-terrain (HAT) lines are vertical lines that run from an entity to the ground and display the altitude of the entity (using whatever units are configured on the Display Units Settings page) (Figure 194). They help you understand the relationship of the entity to the terrain.

HAT lines are shown only for entities that could be above the ground, such as fixed- and rotary-wing vehicles. You can show and hide these lines globally or you can specify that they display only for the selected entities.

As you create or edit an entity, VR-Forces displays HAT lines even if they are otherwise disabled.

Figure 194. Entity height-above-terrain lines



⚠ Displaying height above terrain lines can reduce the frame rate.

➤ **To show or hide height-above-terrain lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 187 on page 472).
3. To display lines, select the Enable Height Above Terrain Lines check box.
4. To display lines only for selected entities, select the Only Display Height Above Terrain Lines For Selected Entities check box.

> In VR-Forces, you can also enable and disable entity HAT lines by choosing **Settings > Entity Height Above Terrain Lines** or by clicking the Entity Height Above Terrain Lines button ( ) on the Display Settings Toolbar.

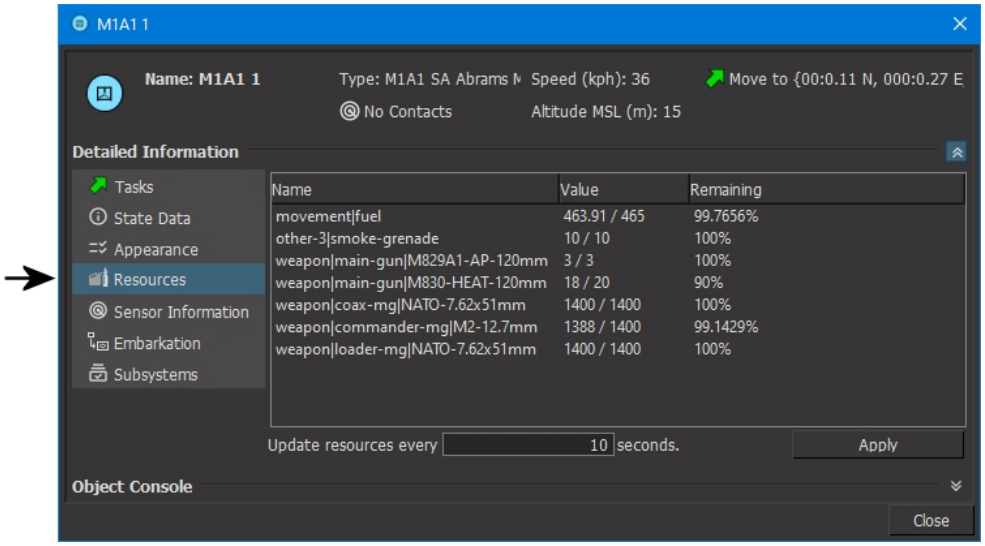
21.7. Displaying Entity Resources

VR-Forces tracks the amount of resources that a simulation object has. You can view the status of resources in the Information dialog box.

21.7.1 Viewing Numerical Resource Data

The Information dialog box lists the simulation object's current amount of resources, its starting amount, and the percentage remaining (Figure 195).

Figure 195. Resource Information page



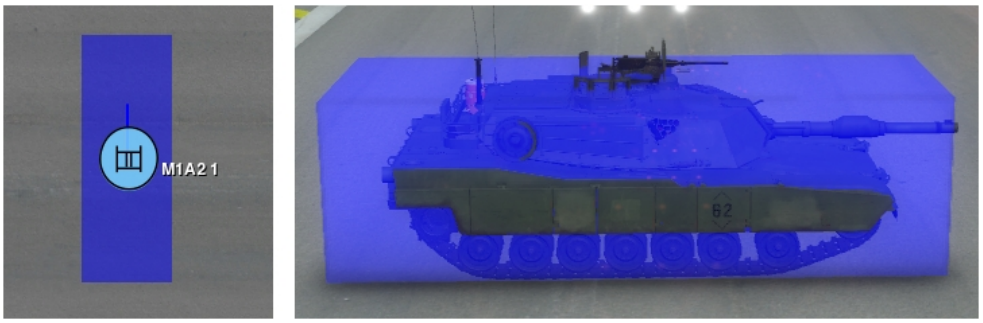
➤ **To view a simulation object's resource status:**

1. Select the simulation object.
2. Choose **Objects > Information**, or press the **I** key.
3. Expand the Detailed Information section.
4. Select the Resources page.

21.8. Displaying Bounding Volumes

You can display bounding volumes for simulation objects. Bounding volumes show the area around a simulation object that the sim engine uses for intersection calculations. Bounding volumes are shown as translucent volumes in 3D and translucent rectangles in 2D (Figure 196). For simulation objects in aggregate-level scenarios, the bounding volume is equivalent to the footprint.

Figure 196. Bounding volumes - 2D and 3D



Bounding volumes for units and bounding volumes for individual entities are managed by different settings. For information about displaying bounding volumes for units, see "19.4. Displaying Unit Icons and Bounding Volumes" on page 448.

21.8.1 Enabling and Disabling Bounding Volumes for Entities

Display of bounding volumes for entities is an observer setting.

➤ **To enable or disable bounding volumes for individual entities:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page.
3. Select the observer mode for which you want to enable or disable bounding volumes.
4. Select or clear the Bounding Volume (Individual Entities Only) check box.

You can also enable and disable bounding volumes for entities by choosing **Observer > Bounding Volumes** or clicking the Bounding Volumes button ( ) on the Observer Settings Toolbar.

21.8.2 Showing Bounding Volumes Only for Selected Entities

You can configure VR-Forces so that it shows bounding volumes only for selected simulation objects.

➤ **To display bounding volumes only for selected entities:**

1. Enable bounding volumes, as described in "21.8.1 Enabling and Disabling Bounding Volumes for Entities" above.
2. Choose **Settings > Display**. The Display Settings dialog box opens.
3. Select the Entity Display Settings page.
4. Select the Show Bounding Volumes for Selected Entities check box.

21.9. Configuring Object Console Messages

The Information dialog boxes for simulation objects and tactical graphics have a console that displays messages sent from the simulation engine, from a simulation object's plan, from other simulation objects, and from scripts. You can set the notification level that controls the display of these messages. You can save them to the clipboard, and you can clear the display. These messages are also sent to the Object Console Summary Panel.

Scripted tasks can ask questions. You can answer these questions from the Object Console Summary Panel.

21.9.1 Setting the Notification Level for Console Messages

Console messages are displayed only if they match or exceed the notification level set for that object. The notification levels are:

- 0 – Only fatal messages are printed.
- 1 – Warning messages and fatal messages are printed.
- 2 – Some diagnostic information is printed with warnings and fatal messages.
- 3 – Verbose information is printed.
- 4 – Debug information is printed.

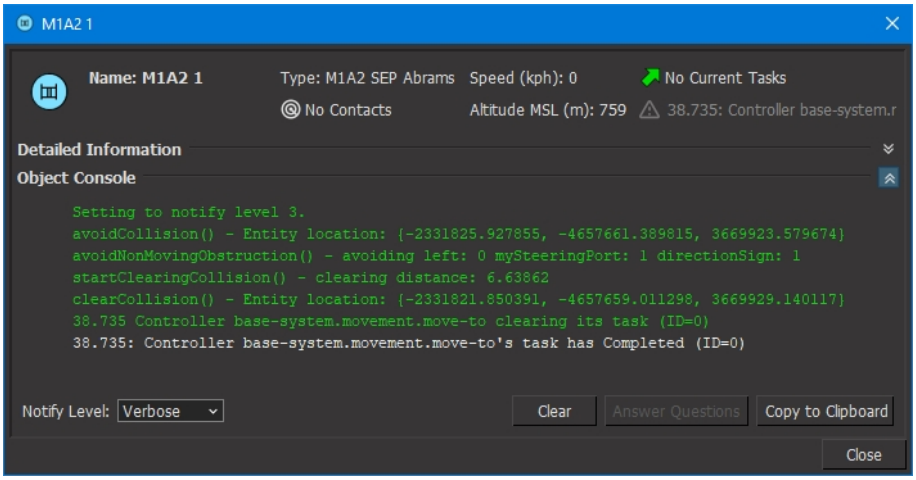
The default notification level is 2.

The notification level is set individually for each object. You can set the notification level in the object's Information dialog box, with the Notify Level set data request (for VR-Forces simulation objects), or with the **objectConsoleNotifyLevel** parameter in *vrfSim.mtl*. (For the set data request procedure, see "40.55. Notify Level" on page 890.)

➤ **To set the notification level for an object's console in the Information dialog box:**

1. Select the simulation object.
2. Choose **Objects > Information**. The Information dialog box opens.
3. Expand the Object Console section (Figure 197).

Figure 197. Information dialog box, Object Console



4. Select a notification level from the Notify Level list.

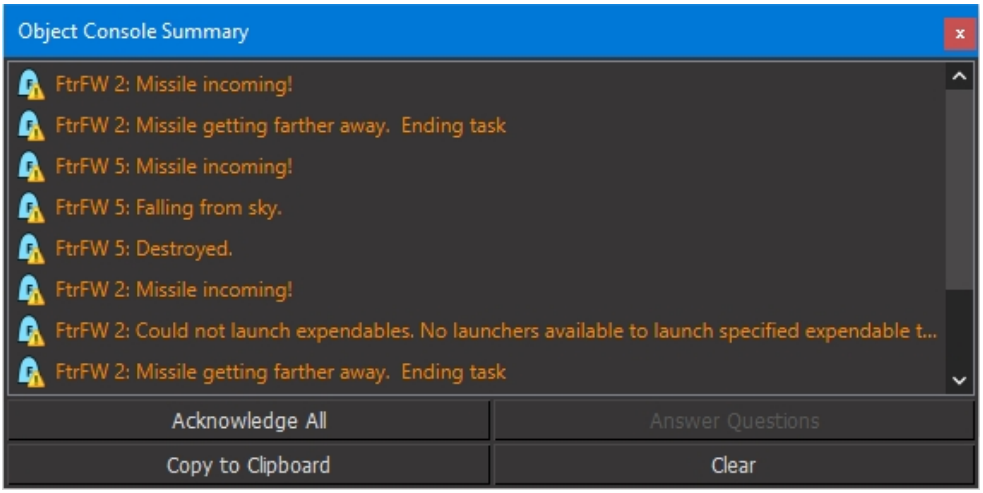
> You cannot use the Object Console to set the Notify Level for remote simulated objects, which are those simulated by another application.

21.9.2 Viewing All Console Messages

You can view all object console messages in one place – the Object Console Summary Panel. Messages are displayed based on the notification levels of the individual simulation objects in the scenario. You can set the notification level in a simulation object's Information dialog box or with the Notify Level set data request.

When VR-Forces sends a message to a simulation object's console, it displays an exclamation point next to the entity until you view the message (unless you disable this feature). The messages in the Object Console Summary Panel also display an exclamation point until you open the Information dialog box for that entity or click Acknowledge All.

Figure 198. Object Console Summary Panel



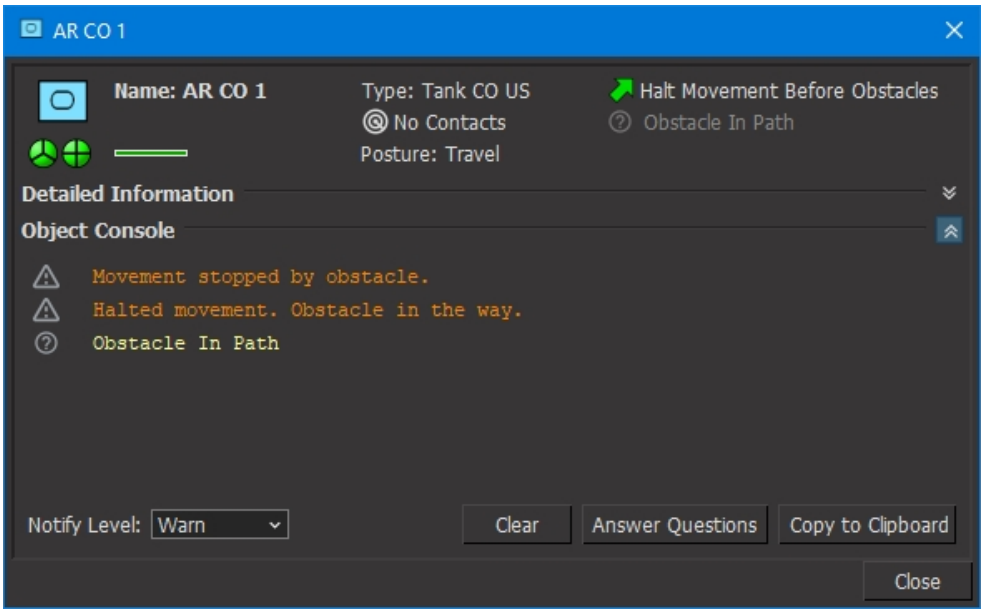
➤ To display the Object Console Summary Panel, choose **View > Object Console Summary Panel**.

➤ To remove the exclamation point from messages in the Object Console Summary Panel, click Acknowledge All.

21.9.3 Answering Questions from Scripted Tasks

Scripted tasks can send questions to the object console. Questions are indicated by a question mark next to the message (Figure 199). The script offers a set of responses as buttons, option buttons, or a list. You can answer questions from the Object Console and the Object Console Summary Panel.

Figure 199. Question in console



➤ **To answer a question from a scripted task:**

1. On the Object Console or Object Console Summary Panel, click Answer Questions. A dialog box with the possible answers displays.
2. Choose an option.
3. Click OK.

21.9.4 Saving Console Messages

If you want to save the messages that are sent to the console, you can copy them to the clipboard and then paste them into a file.

➤ **To save console messages to the clipboard:**

1. Select the simulation object.
2. Choose **Objects > Information**. The Information dialog box opens (Figure 197 on the previous page).

3. If the Object Console is collapsed, expand it.
4. To save all messages in the console, click Copy to Clipboard. To save specific messages, select them and click Copy to Clipboard.
5. Paste the text into the text editor or word processor of your choice.

21.9.5 Displaying an Object Console Warning Icon

VR-Forces can display a notification icon when an object receives a message on its object console (Figure 200). This option is enabled by default, however the default notification level is Warn and objects do not typically get messages at this level. The more likely use for this icon is to set the notification level to Info or lower and then check the console when a notification is received. For details about how to set the notification level, see "21.9.1 Setting the Notification Level for Console Messages" on page 483.

Figure 200. Object console warning icon



Icons are displayed next to the entity icon in the display window, next to the entity's listing in the Objects List Panel, and, if the entity has recently been selected, on the Last Selected Object Panel.

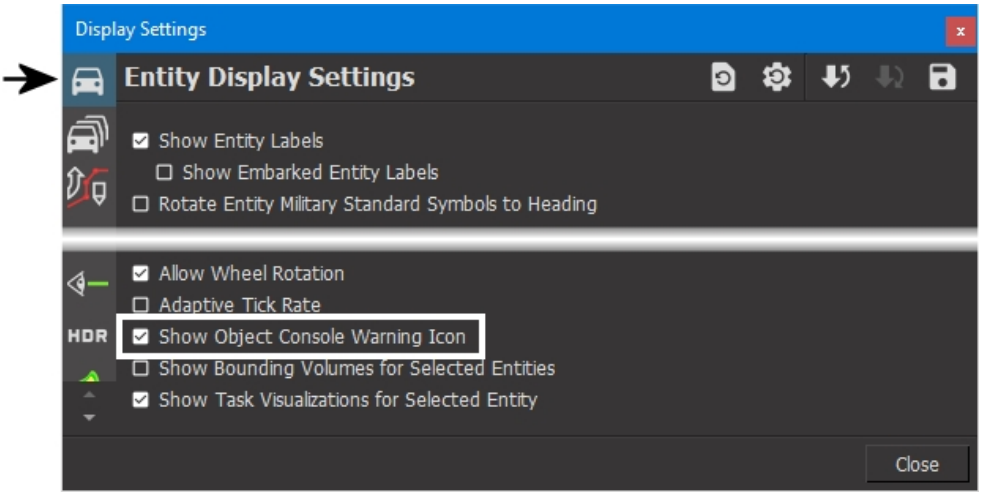
The icon is not displayed if the Information dialog box is open. If an icon is displayed for an object, when you open its Information dialog box, the icon is removed.

If you disable the warning icon, it no longer displays in the display window or the Objects List Panel. It still displays on the Last Select Object Panel, if the entity was the last selected object.

➤ **To enable or disable the console warning icon:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page.
3. Select or clear the Show Object Console Warning Icon check box.

Figure 201. Entity Display Settings, Show Object Console Warning Icon option



21.9.6 Clearing the Object Console

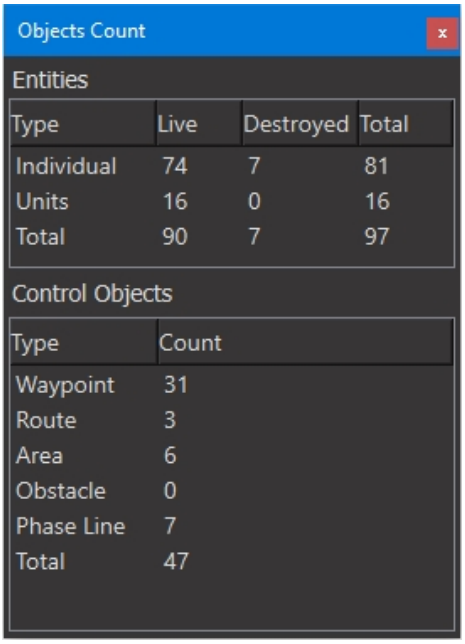
➤ To delete messages from the object console, click Clear.

21.10. Viewing Object Counts

You can display the number of live and destroyed simulation objects in the simulation you are viewing and the number of VR-Forces control objects, by type.

➤ To display the object counts, choose **View > Objects Count Panel**. The Objects Count Panel is added to the display (Figure 202).

Figure 202. Objects Count Panel





22. Creating and Using an Order of Battle

An Order of Battle is a description of the simulation objects and simulation object hierarchy that you can create in a scenario.

22.1. Introduction to the Order of Battle 490
22.1.1 The Order of Battle Panel 490
22.2. Creating an Order of Battle 491
22.2.1 Adding Objects Directly to the Order of Battle 491
22.2.2 Adding Existing Objects to the Order of Battle 493
22.3. Creating Units in the Order of Battle Panel 493
22.4. Instantiating Members of an Order of Battle in a Scenario 493
22.4.1 The Order of Battle Palette 494
22.5. Removing Objects from an Order of Battle 494
22.6. Editing the Objects in an Order of Battle 494
22.7. Changing the Order of Objects in an Order of Battle 495
22.8. Exporting an Order of Battle 495
22.9. Importing an Order of Battle 495

22.1. Introduction to the Order of Battle

In the abstract, an Order of Battle (ORBAT) describes the hierarchical structure of the personnel and resources that are available to a military commander. The term can also be used to refer to the hierarchical structure of the personnel and resources deployed in a particular engagement.

VR-Forces implicitly supports the second sense of the term by listing all simulation objects in a scenario in the Objects List Panel. It also supports the abstract sense of the term by allowing you to construct an Order of Battle that represents a hierarchy of simulation objects that can be instantiated (created) in a scenario.

You create an Order of Battle in the context of a scenario. However, once you create an ORBAT, you can export it and then import it into other scenarios. This lets you quickly create new scenarios that use the same ORBAT for training or scenario development. You can configure VR-Forces so that users do not have access to the Create Object Palette and can only add simulation objects to a scenario if they are in the ORBAT.

An Order of Battle can include any simulation object that you can create from the Create Object Palette. However, unlike the Create Object Palette where each object is essentially a template for creating multiple instances of that object type, each member of an Order of Battle has its own unique ID (UUID). When you instantiate a member of the ORBAT, you only create that one instance of the object type.

In VR-Forces, you can:

- Create an Order of Battle, which involves:
  - Adding members.
  - Removing members.
  - Changing the name or object type of members.
  - Changing the order of members.
  - Organizing members into units.
- Instantiate members of the ORBAT in a scenario.
- Import and export ORBATs.

When you create an ORBAT, it is saved with the scenario.

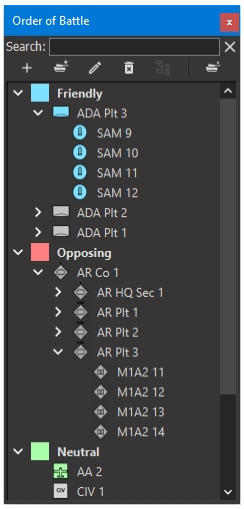
22.1.1 The Order of Battle Panel

You can view and manage the Order of Battle on the Order of Battle Panel (Figure 203). The Order of Battle Panel lists the members of the ORBAT by force.

Each object in the ORBAT list indicates its status in the scenario:

- Gray icon. A member of the ORBAT that is not instantiated in the scenario.
- Colored icon. A member of the ORBAT that is instantiated in the scenario.

Figure 203. Order of Battle Panel



➤ To open the Order of Battle Panel, choose **View > Order of Battle Panel**.

22.2. Creating an Order of Battle

You create an Order of Battle by adding members. You can add any simulation object that is available on the Create Object Palette.

You can add objects to the Order of Battle by either:

- Adding objects to the ORBAT without adding them to the scenario.
- Adding objects that exist in the scenario, but are not currently part of the ORBAT, to the ORBAT.

22.2.1 Adding Objects Directly to the Order of Battle

You can add a simulation object to the Order of Battle, without creating it in the scenario.

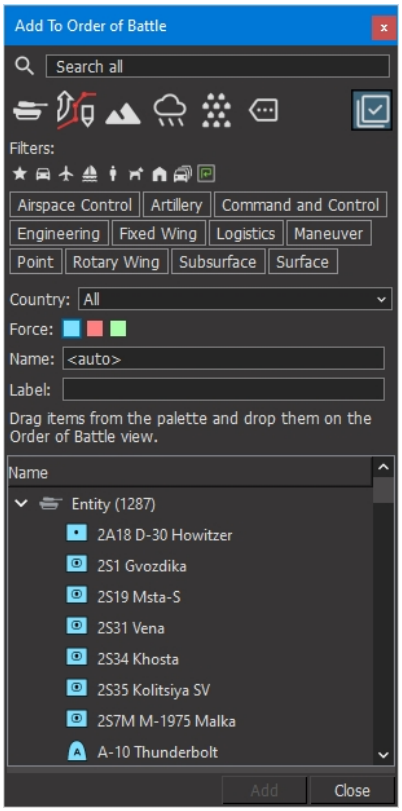
When using this method, if you want to have more than one instance of an object in the ORBAT, you must add a discrete member for each one. For example, if you want to have 10 HMMWVs in the ORBAT, you must add 10 HMMWVs, one at a time.

If you want to add many instances of the same simulation object to the Order of Battle, it may be quicker to create them in the scenario from the Create Object Palette and then add them to the ORBAT.

➤ **To add simulation objects directly to an Order of Battle:**

1. Open the Order of Battle Panel.
2. Click the Add button (+). The Add to Order of Battle dialog box opens (Figure 204). It is similar to the Create Object Palette, except that you can also specify a name and label for the object.

Figure 204. Add to Order of Battle dialog box



3. On the Set Force line, select the force to which you want to add this object.
4. Optionally, filter the list.
5. In the list of simulation objects, select the object you want to add to the ORBAT.
6. Optionally, specify a name for the object.
7. Optionally, add a label for the object.
8. In the Order of Battle Panel, select the force or unit under which you want to add the object. The following rules apply:
   - If you add to a force, it becomes a top level subordinate of the force.
   - If you add to a unit, it becomes a subordinate of that unit.
   - If you add it to an entity, it becomes a sibling of the entity. If the entity is part of a unit, the new member becomes part of the unit.
9. In the Add to Order of Battle dialog box, click Add.

You can quickly add objects to the ORBAT by dragging them from the Add to Order of Battle dialog box to the Order of Battle Panel. The same rules for adding members apply. Also, regardless of the force selected on the Set Force line, when you drag an object onto the Order of Battle Panel, it becomes part of the force you drag it to.

22.2.2 Adding Existing Objects to the Order of Battle

You can create objects from the Create Object Palette and then add them to the ORBAT. You can also use this method to add objects from existing scenarios to an ORBAT. You can add objects individually and you can add all objects at once.

When you add an existing object to the ORBAT, it is added to the same force and its place in the force hierarchy is preserved. If you add a member of a unit, the entire unit is added to the ORBAT.

➤ **To add one or more existing objects to the Order of Battle:**

1. Select the objects that you want to add to the ORBAT.
2. Choose **Simulation > Order of Battle > Add to Order of Battle**.

➤ To add all objects in the scenario to the ORBAT, choose **Simulation > Order of Battle > Add All Simulation Objects to Order of Battle**.

22.3. Creating Units in the Order of Battle Panel

You can aggregate objects listed in the Order of Battle Panel into units. The process and criteria for creating units is the same as described in "18.1.1 Creating a Unit by Combining Existing Simulation Objects" on page 438.

To aggregate members of an Order of Battle into units:

1. Select the objects that you want to aggregate.
2. Choose **Objects > Unit > Aggregate As**, or click the Aggregate As button () on the Order of Battle Panel. The Aggregate As dialog box opens.
3. Select the unit type you want from the list.
4. Optionally, change the order of the unit members.
5. Click OK.

22.4. Instantiating Members of an Order of Battle in a Scenario

If an Order of Battle includes simulation objects that have not been instantiated (created) in the scenario, you can add them to a scenario. Since each member of the ORBAT represents a specific simulation object with its own unique ID (UUID), when you add a simulation object from the ORBAT, you only add that specific object. You cannot add multiple instances of the object type as you would from the Create Object Palette.

➤ **To add a simulation object to a scenario from the Order of Battle Panel:**

1. In the Order of Battle Panel, select the simulation object you want to add to the scenario. (You can only add objects that are not already in the scenario. That is, objects with gray icons.)
2. Click the Add Object button ().
3. Click on the terrain where you want to place the object. The icon for the object in the Order of Battle Panel changes from gray to its force color.

22.4.1 The Order of Battle Palette

There may be cases where you want to set up a training exercise in which participants can only add simulation objects from the Order of Battle. VR-Forces has an Order of Battle Palette that contains the Order of Battle as listed in the Order of Battle Panel, but without any of the controls for adding, removing, or editing the ORBAT. Users can add simulation objects in the same way they would from the Create Object Palette. The Order of Battle Palette is hidden by default, but you can enable it from the View menu.

Using the window layout feature, you can save a layout in which the Create Object Palette is hidden and the Order of Battle Palette is visible. Taking things a step further, you could edit the GUI configuration file so that the Create Object Palette and the Order of Battle Panel are not even available in the GUI. For information about saving window layouts, see "84.2. Saving Window Layouts" on page 1624. For information about configuring the menu options in the GUI, see "Chapter 85. Configuring Toolbars, Dialog Boxes, and Menus" on page 1639.

22.5. Removing Objects from an Order of Battle

You can remove objects from the ORBAT. When you remove an object from the ORBAT, if it is instantiated in the scenario, it stays in the scenario. If you delete an object from a scenario that is a member of the ORBAT, it is not removed from the ORBAT. The object's entry in the Order of Battle Panel changes from its force color to gray.

If you remove a unit, all of its subordinates are removed. If you remove a member of a unit, the rest of the unit is not affected.

➤ **To remove an object from the Order of Battle:**

1. Select the object in the Order of Battle Panel or on the map.
2. Click the Delete button () on the Order of Battle Panel or press the **Delete** key.

22.6. Editing the Objects in an Order of Battle

You can edit the members of an Order of Battle:

- If an object is created in the scenario, editing it opens up the standard Edit dialog box for a simulation object. For details, see "14.10. Editing Simulation Objects" on page 393.
- If an object is not created in the scenario, you can change its name, object type, and force.

➤ **To edit a member of the Order of Battle:**

1. Select an object in the Order of Battle Panel.
2. Click the Edit Object button ().

   If the object has been created, a standard Edit dialog box opens.

   a. Make changes as desired.

   b. Click Update.

If the object has not been created, the Edit Name / Object Type dialog box opens.

a. To change the name, type a different name in the Name box.

b. To change the label, type a different name in the Label box.

c. To change the object type, select a different object in the list.

d. Click OK.

22.7. Changing the Order of Objects in an Order of Battle

You can reorganize the members of the ORBAT. This allows you to populate the ORBAT in the most convenient manner and then organize it into the desired hierarchy. You can move members from units into other units, from units to the force level, and from the force level to units. You cannot change the order in which sibling objects or units are listed.

➤ To reorganize an object in the Order of Battle, in the Order of Battle Panel, drag the object to its new location.

22.8. Exporting an Order of Battle

You can export an Order of Battle so that you can import it into other scenarios.

➤ **To export an Order of Battle:**

1. Choose **Simulation > Order of Battle > Export Order of Battle**. The Export Order of Battle dialog box opens.
2. Type a name for the Order of Battle.
3. Click Save.

22.9. Importing an Order of Battle

You can import an Order of Battle into a scenario. When you import an ORBAT, you can merge it with the existing ORBAT, if any, or you can selectively import members of the ORBAT.

➤ **To import an Order of Battle:**

1. Choose **Simulation > Order of Battle > Import Order of Battle**. The Import Order of Battle dialog box opens.
2. Select the ORBAT you want to import.
3. Click Open. The Order of Battle Selector dialog box opens.
4. To import all members of the selected ORBAT, click Import All.

   To Import selected members of the ORBAT, drag them from the Order of Battle Selector to the Order of Battle Panel.
5. If necessary, close the Order of Battle Selector.



IV. Entity-Level Modeling Concepts

The chapters in this section provide conceptual information about modeling at the entity level.



23. Ground Vehicle Movement

23.1. Overview 500
23.2. How the Move To Task Works 500
23.2.1 Path Planning 500
23.2.2 Move Execution 502
23.2.3 Dynamic Obstacle Avoidance 502
23.2.4 Vehicle Movement On Roads 503
23.3. How the Move Along Route Task Works 505
23.4. How the Follow Entity Task Works 505
23.5. How Terrain Affects Movement 505
23.5.1 Soil Effect on Movement 505
23.5.2 Slope Effects on Movement 508
23.6. Disabling Autonomous Actions 508

23.1. Overview

The ground vehicle movement systems—including ground-tracked, ground-tracks-amphibious, ground-wheels-off-road, and ground-wheels-road—are defined by system definition (*.sysdef*) files. They are configured diferently from one another, but they all adhere to the same format. There are controllers and Lua scripts, which are responsible for executing the movement tasks available to a particular ground vehicle entity, and an actuator that ultimately updates the entity's spatial attributes based on the control input from one of the low-level controllers.

Ground vehicles have three key movement tasks: Move To, Move Along Route, and Follow Entity. There are several other tasks that are based on the key tasks, for example, Patrol Between (points) and Patrol Route.

- Move To. Moves the vehicle to a destination location, causing the entity to first plan a route on roads and around obstacles as appropriate, then following that planned route using Move Along Route and road-movement tasks.
- Move Along Route. Moves the vehicle from point to point along a route tactical graphic, using dynamic obstacle avoidance. Move Along Route does not use road movement (that is, driving in lanes and following traffic), nor does the entity plan paths before moving.
- Follow Entity. Moves the vehicle to ensure that it maintains a station relative to another entity and dynamically avoids obstacles as it moves.

23.2. How the Move To Task Works

The Move To task is designed to move a vehicle from its current location to a given location. While performing the task, entities attempt to use roads, avoid obstacles, use navigation (nav) meshes, and recover from failures as much as possible. The task is implemented as a Lua script using a behavior tree.

The general flow of the script is as follows:

1. A path is planned from the vehicle’s starting location to the destination. In general, this path can be a sequence of road and off-road segments. The off-road segments go around obstacles.
2. The vehicle follows each segment in turn.
    - For off-road segments, the vehicle uses a Move Along Route subtask.
    - For road segments, the vehicle uses a rail-move-along subtask, which drives the vehicle along a road, respecting lane lines and traffic.

23.2.1 Path Planning

Path planning involves several distinct types of logic, depending on the terrain. When roads are available and road driving is desired, the planner automatically looks for a road network. When road movement is not desired, then the planner either uses a nav mesh or creates a clear route between feature obstacles.

Planning a Path Using Roads

When the Move To task tries to plan a route using roads, it uses the VR-Forces feature path planner to find a route on a network of road features. A road feature is any feature defined to be a MAK_ROAD in the *./appData/settings/featureconfig.txt* file. The planner makes a query for road features in the terrain database that are located around the start location and between the start and the end destination. The planner finds the point on the road network closest to the start and then searches to find the point closest to the destination.

The search finds the lowest-cost route on the roads, where the cost of each road is given in the descriptor for the rail-path-controller in the movement system (for example, *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/ground-wheels-road.sysdef*). The road categories in that descriptor are defined in terms of terrain feature attributes in the *terrainconfig.txt* file.

If there are multiple, disjointed road networks between the start and end locations, the path planner tries to find a path that goes along the closest road network, then off road to the next road network, then along that road network, and so on, until it reaches the destination.

Planning an Off-road Path Between Feature Obstacles

When the path needs to go between two locations that are not on a road, VR-Forces examines the path and modifies it as needed to ensure that it does not go through any obstacles that are part of the terrain.

To do this, a path planner gathers obstacle feature data from the terrain in the area between the two locations. The feature attributes that define obstacles are given in the definition of the MAK_OBSTACLES query in *./appData/setttings/featureconfig.txt*. With this set of feature obstacles, the planner checks lines between the two locations, and then between locations and obstacle corners, in different combinations until a clear path is found from the start location to the end destination. Note that this is not an optimal visibility graph algorithm, therefore the resulting path may take a longer way around some obstacles than might be desired.

Path Planning on the Nav Mesh

If a navigation mesh is available, each off-road segment is planned using this nav mesh. The planner takes the costs of different types of soils and the slope of the terrain into account as it plans the path. If navigation mesh planning is not successful—for example, if there is no navigation mesh in the area—then the segment will be planned around obstacles using feature path planning.

Navigation Preferences

Path planning only uses roads if the entity's Navigation Preferences are set to Prefer Roads. This preference is set in the movement system and can be changed with a set data request (see "40.54. Navigation Preferences" on page 890). Wheeled-road vehicles prefer roads by default, while wheeled-off-road and tracked vehicles prefer to Ignore Roads. In any case, if the path starts or ends in a location that is not on a road, that part of the path will be off-road.

23.2.2 Move Execution

While the vehicle moves along its path, it dynamically responds to obstacles.

- When the vehicle is on a road, it uses road driving logic to interact with other vehicles. For details, see "23.2.4 Vehicle Movement On Roads" on the facing page.
- When the vehicle is off road, it uses dynamic obstacle avoidance to avoid other moving entities as well as stationary vehicles and obstacles. For details, see "23.2.3 Dynamic Obstacle Avoidance" below.

In either kind of driving, if the vehicle becomes blocked by obstacles, it backs up a short distance and attempts to drive around the obstacle. If backing up fails several times, the script makes one attempt to restart with planning from its current location.

23.2.3 Dynamic Obstacle Avoidance

As a vehicle moves to a destination location, it continually avoids obstacles. Obstacles include both other entities and obstacle features in the terrain.

- Obstacle features. Obstacle features are found by querying the terrain. This returns obstacles per the definition in *./appData/settings/featureconfig.txt* for MAK_OBSTACLES. The avoidance logic retrieves features a certain distance in front of and to the side of the vehicle, and refreshes its list of obstacles periodically so that it knows about obstacles out to a reasonable horizon.
- Entities. The avoidance logic also gathers a list of nearby entities from the VR-Forces object manager. Because entities can move, it is necessary to look farther ahead and to the side for entities that may be moving toward the vehicle.

The avoidance logic considers future vehicle paths that go straight ahead and turn to the side. For turning paths, a turning radius is approximated, based on the vehicle's speed and turning ability. For each speed, the logic computes speed constraints on the path that allow the vehicle to stop before the obstacle or pass in front of it. The logic then chooses path and speed that allow the vehicle to make the fastest progress toward the destination.

Note that the obstacle avoidance logic is choosing a direction to travel at every instant, but is not planning a path through the obstacles it sees. The avoider also has a limited horizon of obstacles. Therefore, the vehicle could become trapped either by a very large "alley" or by moving entities that close off its path.

23.2.4 Vehicle Movement On Roads

Some terrains have vector road networks. Ground vehicles can use these networks to move precisely through the terrain.

> ⚠ Road driving is valid only for ground vehicles. If you give a movement task with a road driving element (including Treat Route as Road in a Move Along Route task, or Prefer Roads in a spawn pattern's default plan) to any other type of entity, it will fail.

> ⓘ Vehicles follow vector features and ground clamp to the terrain. If there is a crater in the vehicle's path, the vehicle drives through it. If a bridge is destroyed, the road vector remains and the vehicle drives across the terrain that the bridge previously spanned.

The Road Path

When the Move To task begins moving a vehicle on a road, it starts by constructing a travel path through each intersection. The algorithm for path construction attempts to interpret road feature data such that they represent real-world roads and intersections. Road curves and intersections are therefore assumed to have a minimum curve radius that allows a "design vehicle" to drive on them. The design vehicle is a large bus with a turn radius of 8.4 m and a width of 2.4 m. These numbers are hard-coded. As a result, while road intersections may be rendered in the GUI as sections of pavement that intersect at right angles, the vehicle path may be a circular arc that cuts inside the corner of the pavement.

If there are multiple road intersections that are closely spaced, the road driving algorithm attempts to smooth out the travel path to some extent in order to accommodate the design vehicle. If this is not possible, then a travel path with a smaller turn radius may be generated.

Road Driving Behavior

Vehicles driving on roads move strictly along road lanes. Their speed is determined by the road speed limits and curves in the road.

- Initial lane acquisition. Vehicles generally do not start at the center of a lane and oriented to the road. Therefore, their initial movement involves moving into a lane as they move down the road.
- Path control. Vehicles on roads follow the lane centerlines exactly. Vehicles will never move to one side of the lane or the other to avoid an obstacle, for example. At every path vertex where the road changes direction, the vehicles follow a circular arc between segments.
- Lane selection. On multi-lane roads, vehicles prefer not to use the left lane. At intersections where the route turns, vehicles will make sure they are in a lane that connects to the next road. When there is slower traffic ahead of them, vehicles may change lanes to pass, but not if they are near an intersection and the passing lane

does not connect through the intersection.

Vehicles may use an extra "breakdown lane" or "shoulder" that is not specified in the feature data to pass a slow or stopped vehicle ahead, if this option is enabled for the entity. To enable this behavior, the add-breakdown-lanes parameter must be enabled for the entity's movement system in the SOE.

When appropriate, vehicles change lanes. To do this, they follow a smooth arc between lanes. They effectively occupy both lanes while they are changing lanes.

- Obstacle detection. Vehicles look ahead of themselves in their lane to detect other entities. If they detect another entity that is in the lane a certain distance ahead, the vehicle considers that the entire lane is blocked at that point in the road.
- Speed control. Vehicles constantly adjust their speed so that they obey speed constraints at their present position on the road and ahead. The default speed is the entity's ordered speed. Constraints include the nominal speed limit of the road (determined by the size, that is the number of lanes, in the road), the radius of the curves ahead, and the distance to any obstacle ahead. If the vehicle needs to change lanes, its speed may be limited by the distance available to change lanes, since a lane change requires more space when the vehicle is traveling faster.

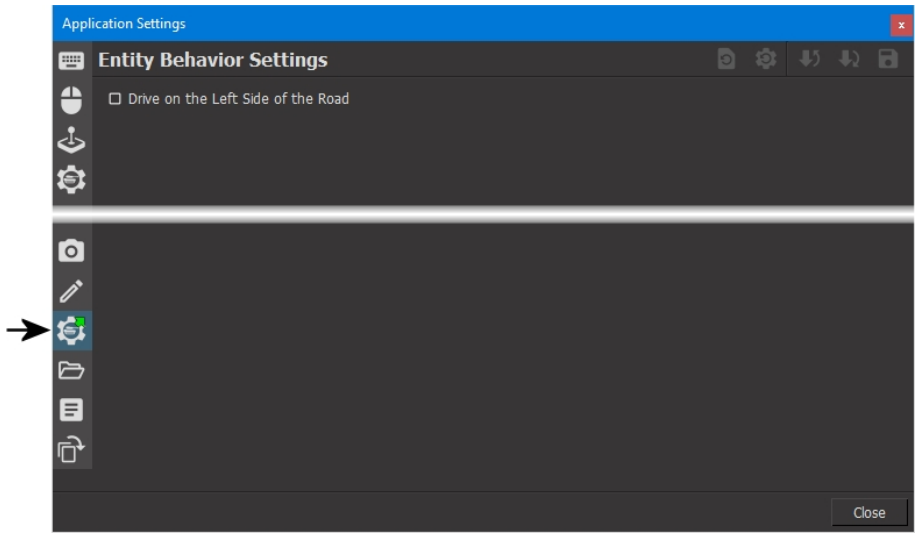
Specifying the Side of the Road to Drive On

By default, entities that use road driving behavior drive on the right side of the road. You can specify that entities drive on the left side of the road. For this setting to take effect, you must make the change before you load a scenario or before you rewind it. If you change the setting after you rewind a scenario or while the scenario is running, the behavior does not change until the next time you rewind the scenario.

➤ **To specify the side of the road to drive on:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Entity Behavior Settings page (Figure 205).

Figure 205. Entity Behavior Settings page



3. Select or clear the Drive on the Left Side of the Road check box.
4. Click Close.

23.3. How the Move Along Route Task Works

The Move Along Route task takes a route object and performs a sequence of movements directly to each vertex of the route. There is no path planning done as it moves toward the next vertex. However, the vehicle does avoid obstacles as it moves. If an obstacle causes the vehicle to move away from the path as it approaches a vertex, it will nevertheless consider that it has reached the vertex if it moves even with the vertex on a parallel path. While the vehicle is moving, its speed is constrained to ensure that it follows the route reasonably well even if the route curves.

23.4. How the Follow Entity Task Works

The Follow Entity task causes the vehicle to move towards a location that is offset from another entity by a specified distance behind and to the side. As the lead vehicle moves, the location of this offset station also moves, so the following vehicle has a continually changing destination. The following vehicle does not do any path planning, but it does avoid obstacles dynamically as it moves.

23.5. How Terrain Affects Movement

Two factors that can impact ground entities' movement are the type of soil and the slope of the terrain.

> There is a configuration option in *navigationProfiles.mtl*, `soil-types-to-tag-with-surface-char`, which lists soil types. Only soil types included in the list will generate navigation tags specific to that soil type. Other soil types receive the default navigation tag. For details, see that section of the file.

23.5.1 Soil Effect on Movement

In VR-Forces, the movement of simulation objects is affected by the surfaces on which they are moving. The factors that affect vehicle movement for specific soil types are configured in the `soil-list` block in a simulation object's *.entity* file or in its movement system definition file.

Table 26 lists the soil types supported for terrain databases and how they map to the roughness types you can configure in the `soil-list` block for VR-Forces simulation objects.

Table 26. Soil type/roughness type mappings

Soil type	Roughness type
ocean	deep-water
deepLake	deep-water
shallowLake	shallow-water
deepRiver	deep-water
shallowRiver	shallow-water
shallowPond	shallow-water
shallowStream	shallow-water
dryGround	hard-packed
pavedRoad	paved-road
gravelRoad	gravel
dirtRoad	gravel
asphalt	hard-packed
USRailroad	hard-packed
softSoil	sand
swamp	muck
mud	muck
forest	hard-packed
grass	hard-packed
cultivatedFields	hard-packed
orchards	hard-packed
rock	rocks
boulder	rocks
sand	sand
building	hard-packed
tree	hard-packed

You can change the way a ground vehicle moves on each soil roughness type by editing the appropriate movement system definition file in *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement*:

- *ground-tracked.sysdef*
- *ground-tracks-amphibious.sysdef*
- *ground-wheels-off-road.sysdef*
- *ground-wheels-road.sysdef*

Near the end of each these movement sysdef files, inside the **meta-data-entry-list** block, there is a **soil-list** block that lists the soil roughness types (paved-road, hard-packed, gravel, and so on). Here you can adjust the **acceleration-factor** and **stopping-factor** values for each roughness type; VR-Forces uses those values to calculate how fast a vehicle can accelerate, brake, and turn on that type of surface. (Do not modify the **max-speed-factor** value; it is not used and changes have no effect.)

The **acceleration-factor** affects the surface's drag on a vehicle. Use values ranging from 0 to 1, where lower values increase drag. A value of 1.0 means the surface exerts no drag (a paved road, for example); a value of 0.0 means the surface's drag prevents the vehicle from moving at all (deep water).

The **stopping-factor** affects the traction the vehicle has on the surface. Use values ranging from 0 to 1, where lower values reduce traction. A value of 1.0 means the surface provides maximum traction (a paved road, for example); a value of 0.0 means the surface provides no traction (such as muck).

For example, to improve a road vehicle's ability to accelerate in snow, you might increase the **acceleration-factor** in the snow section from

```
(meta-data-entry-list
 (soil-factors
  (soil-list)
   ...
   (snow
    (acceleration-factor 0.050000) ; for deep snow
```

to

```
   (snow
    (acceleration-factor 0.080000) ; for deep snow
```

In addition, you can modify these parameters to control the way weather affects ground vehicle movement over different soils:

- **deep-snow-depth**:
  Depth of snow for full mobility degradation effect; in meters. Default is 1.0.
- **saturation-water-depth**:
  Rain accumulation for full wet effect; in meters. Default is 0.2.
- **water-slide-depth**:
  Depth of accumulated rain that makes a hard surface slippery; in meters. Default is 0.02.
- **water-slide-factor**:
  Fraction of traction retained on a hard, wet surface. Choose a value in the range 0–1. Default is 0.65.
- **hard-freeze-temperature**:
  Temperature at which shallow water freezes enough to support a vehicle; in Celsius. Default is -5.0.

You can find the parameters in *./data/simulationModelSets/EntityLevel/physicalWorldParams.mtl*.

23.5.2 Slope Effects on Movement

The slope of the terrain has several effects on ground movement.

- For planning paths in nav meshes, higher slopes cost more (whether the slope is up or down). In the movement *.sysdef* files in *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement*, the `slope-avoidance-factor` in the `navigation-preference-controller` descriptor can vary the effect.
- Ground slope is not taken into account by the path planner that plans around feature obstacles.
- In the vehicle dynamics model, moving up slope requires a higher throttle setting than moving over level ground, and moving down slope requires a lower setting. On the best soil surface (dry pavement), vehicles can just barely move up the `max-slope` defined in the entity parameters by using maximum throttle. It is possible that vehicles may slide down slopes, especially if the soil is slippery.

23.6. Disabling Autonomous Actions

If you use the Autonomous Actions Enabled set data request to disable an entity's independent intelligent behavior, then all automation is disabled.

For ground vehicle movement, this means that the entity will not plan paths over road networks, between feature obstacles, or in navigation meshes. The vehicle also will not avoid obstacle dynamically as they move. A Move To task causes the vehicle to proceed in a straight line over the terrain to the destination, moving through any obstacles in its way.

When autonomous actions are enabled and the ground vehicle is embarked or the destination is an embarked object, all path planning is disabled. Dynamic obstacle avoidance continues to work as normal, unless you disable autonomous actions.

See "40.3. Autonomous Actions Enabled" on page 866 for more details on this set data request.



24. Aircraft Behaviors

While aircraft in entity-level scenarios can respond to the same tasks and set data requests as ground vehicles, the way they perform actions or movements varies. This chapter provides information on how aircraft behave in entity-level scenarios.

24.1. Fixed-Wing Entity Behaviors 510
- 24.1.1 Placement of Newly Created Fixed-Wing Entities 510
- 24.1.2 Fly Task Behavior Is Different from Move Task Behavior 510
- 24.1.3 Specifying and Maintaining Altitude for Fixed-Wing Entities 510
- 24.1.4 Fixed-Wing Entity Movement on the Ground and in the Air 511
- 24.1.5 How Fixed-Wing Entities Take Off 511
- 24.1.6 How Fixed-Wing Entities Land 512

24.2. Rotary-Wing Entity Behaviors 513

24.1. Fixed-Wing Entity Behaviors

This section describes the behaviors of fixed-wing entities supplied with VR-Forces and the issues you should keep in mind when assigning tasks and set data requests.

24.1.1 Placement of Newly Created Fixed-Wing Entities

When you create a fixed-wing entity, it is placed on the ground. If you want a fixed-wing entity to start a scenario in the air, you must set its altitude at the beginning of the scenario. It will loiter at the specified altitude.

24.1.2 Fly Task Behavior Is Different from Move Task Behavior

Fixed-wing entities can move in response to Fly tasks (Fly Altitude, Fly Heading, and so on) and Move tasks (Move To, Move Along Route, and so on). There are important differences between how fixed-wing entities respond to the different classes of movement tasks. Fixed-wing entities can move on the ground (taxi), in which case they move like ground vehicles.

For Move and Patrol tasks, if a fixed-wing entity is in the air, when it arrives at a point or the end point of a route, it loiters.

Fly tasks are only for entities that are in the air. You cannot use them to cause an entity to take off or land. For Fly tasks:

- Entities move to a new altitude at the specified heading, at the specified climb or descent rate.
- When they reach the new altitude, they continue flying at their current heading.

Given these differences, if you want to change heading, altitude, or both for an entity that is in the air and not executing a Move or Patrol task, use a Fly task. These tasks are not designed to have an entity move to a particular location or along a route. To have an entity move to a particular location or along a route, use a Move or Patrol task. In most cases, Fly Altitude will be preferable to Move to Altitude.

24.1.3  Specifying and Maintaining Altitude for Fixed-Wing Entities

You can explicitly change a fixed-wing entity's altitude:

- The Altitude set data request moves an entity immediately to the specified altitude.
- The Move to Altitude task causes an entity to spiral to the specified altitude at the same coordinates that it had when it received the task.
- The Fly Altitude and Fly Heading and Altitude tasks cause the entity to move smoothly to the new altitude while continuing to fly at its current or newly specified heading.

An entity's altitude changes implicitly if it is given a Move or Patrol task in which the altitude of the target point or vertex is different from the entity's current altitude. When moving along a route, the entity changes altitude as it flies along a line between vertices.

When you assign a task that includes a location and you specify the location by clicking on the terrain, the default altitude is zero. If you do not specify a non-zero altitude in the task dialog box, VR-Forces uses the altitude of the entity at the time the task is assigned. This prevents you from inadvertently specifying a location that would cause the entity to crash.

24.1.4 Fixed-Wing Entity Movement on the Ground and in the Air

Fixed-wing aircraft support takeoff and landing. Since they can move on the ground and in the air, you need to understand their behavior in these different environments. Table 27 and Table 28 describe how fixed-wing entities respond to Move To, Move Along Route, and Patrol Route tasks depending on their current location. The rest of this section provides additional details about takeoff and landing.

Table 27. Fixed-wing entity Move To behavior

If entity is on/in the:	and waypoint is on/in the:	the entity:
Ground	Ground	taxis to point.
Ground	Air	takes off towards the point.
Air	Air	flies to point.
Air	Ground	crashes.

Table 28. Fixed-wing entity route-following behavior

If entity is on/in the:	and next vertex is on/in the:	the entity:
Ground	Ground	taxis to vertex.
Ground	Air	takes off toward the vertex.
Air	Air	flies to the vertex.
Air	Ground	tries to land at the vertex.

Fixed-wing entities have a look-ahead capability. If a route segment goes underground, a fixed-wing entity tries to maintain altitude above the terrain rather than following the route into the ground. This is not nap-of-the-earth (NOE) flight. The aircraft looks ahead a configured distance in its current flight direction. If any terrain blocks the path between it and its destination vertex on the route, it ascends to avoid the terrain. When no blocking terrain is found, it resumes flying toward the destination vertex directly.

24.1.5 How Fixed-Wing Entities Take Off

You can have fixed-wing aircraft take off using the Takeoff On Runway task (for details, see "30.50. Takeoff On Runway" on page 620) or by using innate behavior. This topic describes takeoff using innate behavior.

Fixed-wing entities take off by moving to a waypoint that is above ground level or by following a route that has a vertex above ground level.

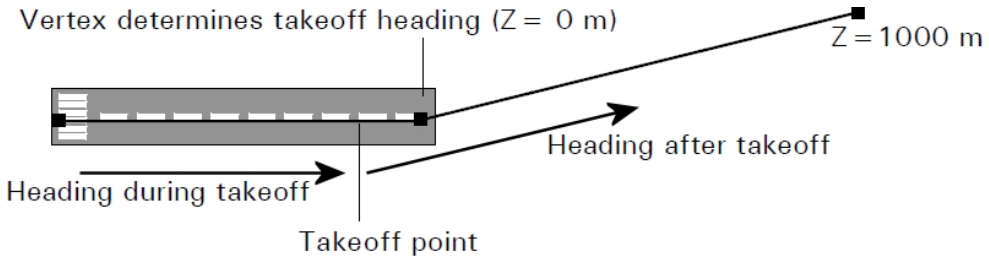
When you task a fixed-wing entity to move to a waypoint that is above ground level, the entity orients itself directly towards that point and accelerates in the direction of the point. When it reaches its minimum lift speed, it takes off and rises to the altitude of the point it is moving to. You can control the distance required to take off by setting the **max-thrust parameter**.

When a fixed-wing entity takes off, it does not take into account the soil type or terrain, except that if the slope of the terrain along which the entity must move to take off is greater than the **max-slope** parameter, the entity halts.

When you create a waypoint towards which an entity will take off, or create a route along which an entity will take off, you are responsible for ensuring that the entity is properly oriented towards the runway.

If an entity takes off as part of route following, you can define a runway with two points on the ground. If an entity detects that the point in the route after the one it is moving towards is in the air, it uses the current point as the take-off direction, but takes off as soon as possible and flies to the point in the air. Figure 206 illustrates this behavior.

Figure 206. Defining a runway for take-off



24.1.6 How Fixed-Wing Entities Land

You can have aircraft land using a fixed-wing landing task (see "30.17. Land At Airport" on page 596, "30.18. Land at Airport Near Location" on page 596, "30.19. Land On Route" on page 596, or "30.20. Land On Runway" on page 597), or as described in this section. Using a fixed-wing landing task automates the process of setting up the approach route and landing.

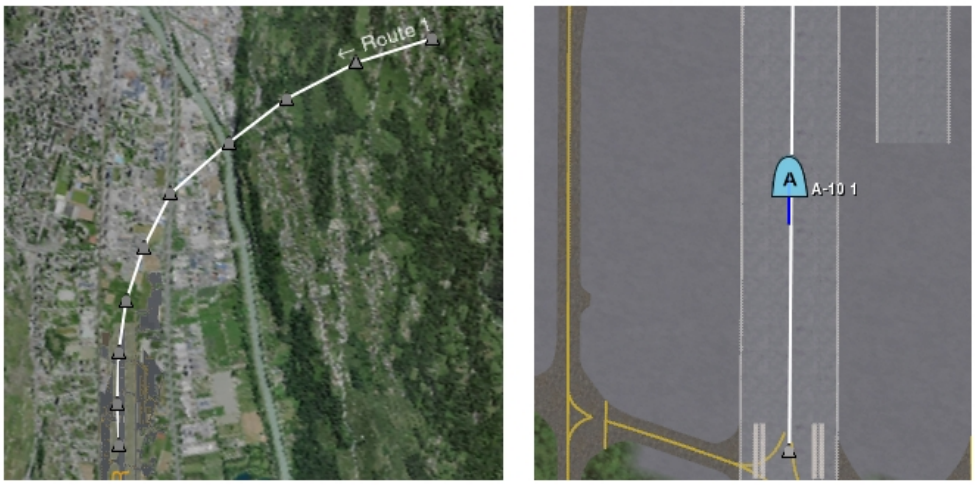
When a fixed-wing entity follows a route and determines that the next vertex in the route is on the ground, it decelerates as it reduces altitude towards that vertex. If it does not have enough distance from the previous vertex to decelerate sufficiently, it may crash. To ensure adequate deceleration, you may want to create a trigger that sets the entity's speed when it approaches the runway.

> Any time an entity is within the **on-ground-tolerance** distance of the terrain and it is moving within 10 percent of its minimum lift speed, it will try to land. The **on-ground-tolerance** for most fixed-wing entities is 10 meters.

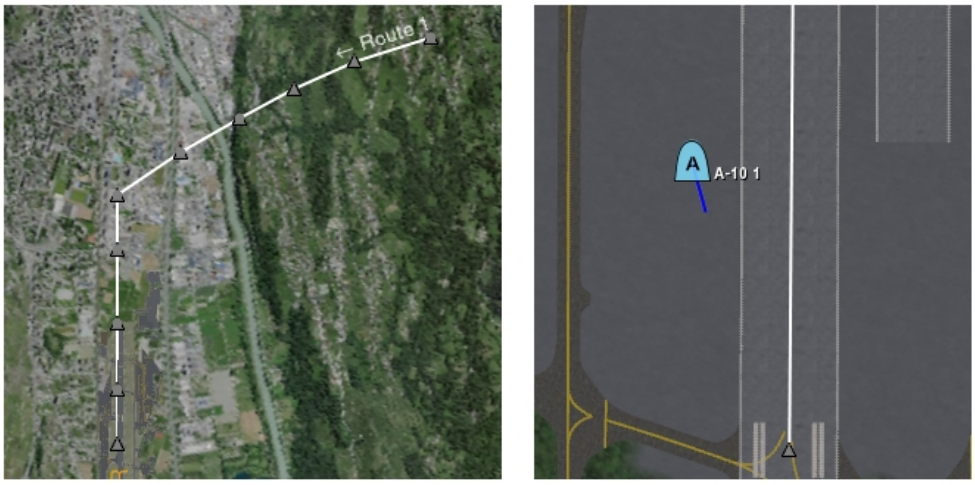
To orient an entity for landing, you may have to create several vertices prior to the runway to properly line up the entity. Figure 206 shows two different routes ending at an airport (the routes have been edited to improve visibility). The first route has vertices that

line up the entity with the runway. The second route curves in towards the runway, so the entity does not approach the runway in a straight line. Therefore, as it reaches the landing point, it is not lined up with the runway and it lands on the taxiway.

Figure 207. Landing fixed-wing entities



Entity lands on runway



Entity lands beside runway

24.2. Rotary-Wing Entity Behaviors

The rotary-wing entities supplied with VR-Forces have certain behaviors that affect how they perform tasks and respond to set data requests.

Rotary-wing entities are created on the ground unless you specify an altitude at creation. (For details about specifying an altitude when you create an air platform, see "14.4. Specifying an Object's Altitude" on page 386.)

If a rotary-wing entity is airborne and it is not assigned a task, it hovers at the specified location and altitude. When it completes a task, it hovers in the location where it completed the task.

The `terrain-check` parameter determines how a rotary-wing entity interacts with the terrain. If `terrain-check` is False, a rotary-wing entity ignores the terrain. It could appear to fly below ground. However, when `terrain-check` is set to False, VR-Forces does not have to check ground points, so performance may improve.

When `terrain-check` is True, rotary-wing entities interact with the terrain. If a rotary-wing entity approaches a ground point at three meters per second or less, it lands. If it approaches the ground at a greater speed, it crashes.

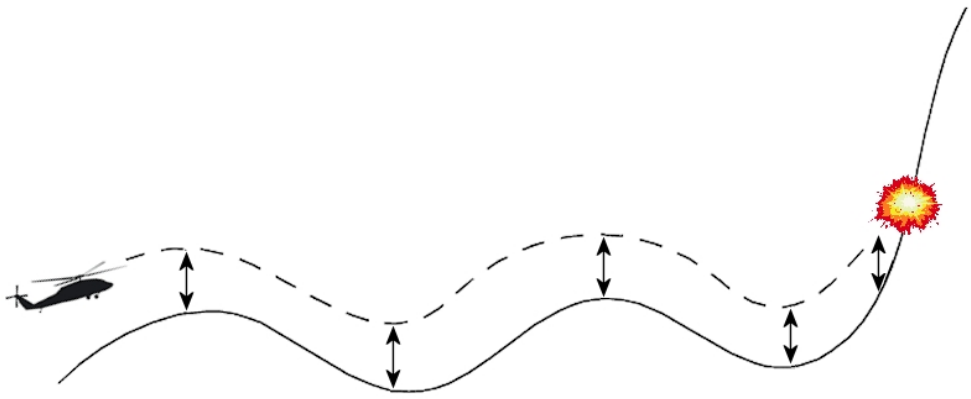
When you assign a task that includes a location and you specify the location by clicking on the terrain, the default altitude is zero. If you do not specify a non-zero altitude in the task dialog box, VR-Forces uses the altitude of the entity at the time the task is assigned. This prevents you from inadvertently specifying a location that would cause the entity to crash.

When following a route, rotary-wing entities attempt to reach the altitude of each vertex along the route. If they arrive at the vertex location before reaching that altitude, they ascend or descend until they reach the altitude of the vertex.

Helicopters have a basic terrain avoidance algorithm that helps prevent crashing if they are sent to waypoints or follow routes that do not have an altitude. The terrain avoidance logic maintains a distance above the ground; it does not look ahead. Therefore, a helicopter could crash if the terrain rises suddenly (Figure 208), such as a steep hill or building. You can view the terrain profile of the route to check for intersections with the terrain. (For details, see"65.1.1 Displaying a Line's Terrain Profile" on page 1267.)

Rotary-wing aircraft take wind resistance and the weight of the aircraft, including the weight of any embarked objects, into account when calculating fuel consumption. If an aircraft's power plant appearance is set to off and it is tasked to move, the rotors spin up and the aircraft will not move until they reach full speed.

Figure 208. Helicopter terrain avoidance





25. Unit Behavior

This chapter describes how units are modeled in entity-level scenarios.

25.1. How Units Are Organized 516
25.1.1 Reorganizing Units 516
25.2. How Units Move 517
25.2.1 Unit Formations 517
25.2.2 Ground Vehicle Unit Movement 518
25.3. How Units Carry Out Tasks 518

25.1. How Units Are Organized

The organization within a unit is based on the echelon ID of each member. For example, in Table 29, four unique entities (identified by name) are assigned designators in **2 Plt**.

Table 29. Example of names and echelon IDs

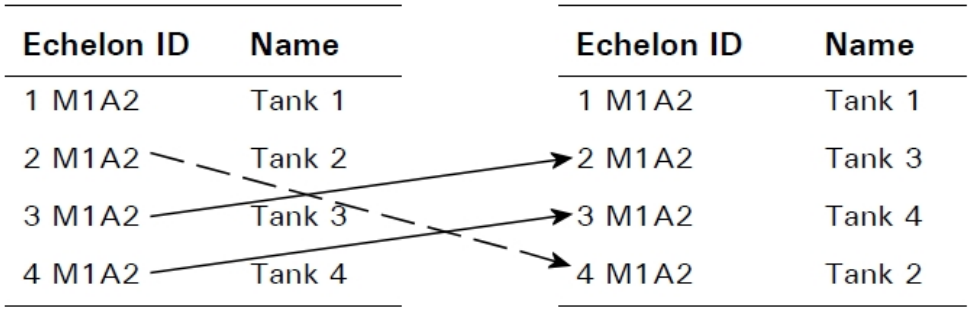
Name	Echelon ID
Tank 1	1 M1A2, 2 Plt, 1 Force
Tank 2	2 M1A2, 2 Plt, 1 Force
Tank 3	3 M1A2, 2 Plt, 1 Force
Tank 4	4 M1A2, 2 Plt, 1 Force

Assignment of echelon IDs is handled automatically by VR-Forces when you create a unit.

25.1.1 Reorganizing Units

Reorganization is the reassignment of echelon IDs to the members of a unit when a member is destroyed. When reorganization is enabled, if a member of a unit is destroyed, the destroyed member is assigned the highest numerical ID in the unit, and the surviving members' echelon IDs are moved down numerically by one. For example, assume a platoon as shown in the leftmost table in Figure 209. If **2 M1A2** (currently the entity named Tank 2) is destroyed, reorganization would result in the new echelon ID assignments illustrated in the rightmost table. The former **3 M1A2** is now **2 M1A2**, former **4 M1A2** is now **3 M1A2**, and the former **2 M1A2**, is now **4 M1A2**. Their echelon IDs have changed. Their names and object IDs have not.

Figure 209. Reorganization of Platoon 2



You can configure VR-Forces to reorganize units automatically, or you can reorganize manually. Manual reorganization does not require you to reassign each designator; you simply order the unit to reorganize with the Reorganize command.

25.2. How Units Move

When a unit receives a movement command, VR-Forces calculates the path the unit leader must follow to reach the destination. It then calculates offset destinations for each subordinate in the unit as well as the routes to those destinations, which are offset from the leader's route. The offset destinations are subordinate positions at the group destination, based on the unit's current formation. You can find details about how units perform movements for each task in "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583.

25.2.1 Unit Formations

When you create a unit from the Create Object panel, the entities are placed in the default formation defined for that unit. To learn about these formations, see "71.5. Editing Formations" on page 1411.

You can specify the unit's formation using any of these:

- Move Into Formation task. See "30.26. Move Into Formation" on page 603.
- Transition Into Formation task. See "30.52. Transition Into Formation" on page 621.
- Formation set data request. See "40.33. Formation" on page 879.

Closing Formation

VR-Forces can close up holes in formations that are created when a unit member is removed from a formation. When a unit is ordered to form up, it assigns formation positions to each of its members. Each position in the formation has an internal ID. If at some point one or more of the unit's members are destroyed, removed, or independently tasked, the unit reassigns the formation IDs for its list of members. Members that are removed, destroyed, or independently tasked are not given formation IDs. The active members are moved up in the formation to fill the vacancies. Assignment to a different formation position has no effect on echelon designator.

The process of closing formation is internal to a unit. Formation position is not reported as part of simulation object information. The only way to determine a simulation object's formation position is to observe the formation as the unit moves.

Closing formation is an inherent feature of units in the VR-Forces application. You do not need to do anything to put it into effect and you cannot turn it off.

Closing Formation Versus Reorganization

Closing the formation of the entities in a unit is not the same thing as reorganization. Reorganization changes the echelon designators of its members. Closing formation does not change the echelon designators of unit members, or the tasks a simulation object is assigned. It only affects the relative positioning of members within a formation.

25.2.2 Ground Vehicle Unit Movement

Key Ground Vehicle Unit Movement Controllers

The primary movement system for ground units is the ground-disaggregated-movement system. It is responsible for implementing movement tasks at the unit level, which ultimately sends tasks to its individual subordinates.

Ground Movement Tasks

The ground vehicle movement tasks build on one another: the higher level tasks use the lower level tasks and also implement more complex behaviors. Many of the higher level tasks use the Lua API's Behavior Engine.

25.3. How Units Carry Out Tasks

Like other simulation objects, each unit has a plan and you can assign independent tasks to units. When you give a task to a unit, that unit sends radio messages to its subordinates directing them to carry out their role in the task, for example, getting into formation and moving to a waypoint.

The precise way in which a unit sends out messages to its subordinates is part of the modeling of the unit that is built into the VR-Forces application.



26. Suppression and Damage

26.1. Suppression 520
26.1.1 Suppressive Fire Tasks 520
26.1.2 Effects of Suppressive Fire 520
26.1.3 Configuring Suppression Effects 521
26.2. Multi-hit Actuator 522
26.2.1 Damage Points 522
26.2.2 Damage System 522

26.1. Suppression

Suppressive fire in VR-Forces is when an individual entity or subordinates of a unit fire near a target to reduce the combat effectiveness of entities near the target.

26.1.1 Suppressive Fire Tasks

VR-Forces has suppressive fire tasks:

- Provide Suppressive Fire is available to individual entities. See "31.31. Provide Suppressive Fire" on page 645.
- Suppressive Fire is available to vehicle units. See "33.7.8 Suppressive Fire" on page 678.
- Fire Team Suppress Target is available to infantry fire teams. See "31.19. Fire Team Suppress Target" on page 637.

Fire is rapid at first, then sustained for a period of time and maximum rounds to fire, or until the entity runs out of ammunition. The rapid and sustained rates of fire are configured in the weapon's system definition file. If you want to suppress a floor of a building, set the altitude at or slightly above the altitude of the floor of the building. Suppressive fire is executed even if line of sight is blocked near the target location, so you can place the target in a building and have fire on the wall in front of it.

To learn how entities are affected when they are fired upon, see "26.1.2 Effects of Suppressive Fire" below.

26.1.2 Effects of Suppressive Fire

Suppressive fire insults on an entity are accumulated as part of the internal state of the entity. This suppression value also continually decays over time. Therefore large insults, or many small insults in a short time, raises the suppression value, but the value naturally drops back down to its starting level.

An entity is considered suppressed when its suppression value rises above a certain threshold. The entity is no longer suppressed when its decaying suppression value falls below a different, lower threshold. This allows the model to produce a delay in suppression recovery and at the same time continue handling new insults.

The model represents three suppression levels (in addition to "not suppressed") so that behavior can be changed for more severe suppression. The number of suppression levels was chosen arbitrarily. Higher levels of suppression are reached when the suppression value passes higher thresholds. The State Data page in the Information dialog box displays the suppression level of the entity, a value between 0 and 3.

Suppressive fire causes humans that are near the detonation locations to temporarily stop shooting and lose some ability to sense entities. When an entity is suppressed:

- Its sensors may have their sensitivity reduced.
- Its guns may be prevented from firing.
- It may be forced into a crouching posture.
- It may be forced to stop moving and go prone.

The default configuration for humans is:

- Visual sensor sensitivity is reduced to 0.
- Guns are disabled.
- Level 1 suppression forces a crouch.
- Level 2 and 3 suppression stops movement and makes the entity go prone.

The Lifeform Suppression system enables two reactive behaviors. The first one makes the human adopt a crouching posture when it is suppressed. This does not interrupt other movement. The second reaction makes the entity go prone when it is suppressed to level 2 or 3. This interrupts any ongoing movement task. Both reactions restore the original posture when the reaction ends.

Several vehicles are configured so that they cannot fire their machine guns if they are suppressed.

An entity is susceptible to suppression if it has a suppression system. Lifeforms can have a Lifeform Suppression system. Vehicles can have a Vehicle Exposed Crew Suppression system. For details about configuring suppression, see "26.1.3 Configuring Suppression Effects" below.

26.1.3 Configuring Suppression Effects

Suppression effects are configured by parameters in several components.

The suppression system contains a suppression actuator with several parameters to control its behavior.

- **suppression-insult-level**. The level of insult necessary to suppress the entity, that is, reach the level 1 threshold. This value should be set appropriately for the amount of insult caused by munitions, as defined in the damage-model portion of this descriptor. You can set this parameter in the Simulation Object Editor.
- **suppression-recovery-time**. The time required to recover from the level 1 suppression threshold to a not-suppressed state. This same time is required to recover to level 2 from level 3, and to level 1 from level 2. You can set this parameter in the Simulation Object Editor.
- **passing-rounds-suppress**. Determines whether rounds passing by the entity, but not necessarily detonating near it, are considered when detecting insults. You can turn off this effect to save computation time. Default: true.
- **max-detonation-distance-for-passing-rounds**. The maximum distance that the actuator looks for detonations. A line of fire that passes near the entity may detonate some distance beyond the entity. To find such lines, the actuator must look at distant detonations. This may result in the actuator processing many detonations. (The detonation manager makes a distance test for each entity for each detonation, so not much computation is saved by reducing this parameter value.)
- **hostile-fire-only**. If true, the actuator does not consider detonations and passing rounds as insults if they are from a friendly force. Default: true. While the default may be unrealistic in the case of HE shells landing near the entity, it avoids unintended suppression when nearby friendly entities are firing at the enemy.

The actuator descriptor includes a damage-model section that is just like the damage-model in a damage system. For each munition there are two parameters:

- **suppression-detonation-insult**. The maximum insult that this munition can cause, if it detonates at the entity location or passed directly over the entity.
- **suppression-detonation-range**. The maximum range at which the munition can have effect. The insult from the munition decreases linearly until it is 0 at the maximum range.

In ballistic gun controllers, **disabled-by-suppression** indicates whether the gun is susceptible to suppression. If true, the gun is not allowed to fire while the entity is suppressed. This is true for the handheld weapon systems, but it can also be selectively set for guns on vehicles. For example, tank machine guns are often operated by an exposed crew member, and could be suppressed by small arms fire even if the tank is otherwise unaffected by such fire. The handheld systems have this parameter set True, while other machine guns use a variable and so the parameter can be set in the Simulation Object Editor.

In sensor components, **suppression-degrades-sensor-performance-by** indicates how they are affected by suppression. It is a value from 0 - 1.0. A value of 0 means that the sensor is not degraded at all by suppression, while a value of 1.0 means that sensitivity is reduced to 0 by suppression. The value can be set in the Simulation Object Editor.

26.2. Multi-hit Actuator

Some surface watercraft have the Surface Multiple Hit Damage system. The damage system is specified for entities in the Simulation Object Editor (SOE) on the Systems tab.

26.2.1 Damage Points

The damage state sent on the network is determined by the percent damage, which is calculated by dividing the current damage points by the total damage points. You can configure the values for an entity:

- Total Damage Points. The amount of damage the entity can sustain before it is destroyed. The default values are based on the entity's size. The values are relative. For example, smaller watercraft that can sustain multiple hits, such as Corvettes, have a value of 15, while destroyers are set to 35.
- Percent Damage for Slight Damage State. The percent of damage that the entity must sustain before it is considered slightly damaged.
- Percent Damage for Moderate Damage State. The percent of damage that the entity must sustain before it is considered moderately damaged.

The entity's percent of damage state displays as a horizontal bar in the Information dialog box and the Last Selected Object panel.

26.2.2 Damage System

The Surface Multiple Hit Damage system itself measures by Damage Points. The amount of damage points taken by the surface watercraft is a random number between the maximum and minimum damage points at the level of explosive power (kPa) that the

attacking entity dealt. This ensures a level of variability, rather than a flat amount of damage.

Firepower kill, mobility kill, and sensors kill use the probability of effect model. Damage points take the place of the catastrophic kill effect.

For details on configuring damage systems, see "Chapter 78. Configuring Damage Systems" on page 1545.



V. Aggregate-Level Modeling Concepts

The chapters in this section provide conceptual information about modeling at the aggregate level.



27. Aggregate-Level Modeling

This chapter describes the aggregate warfare model and how simulation objects created using the aggregate-level SMSs work (*AggregateLevelBase.sms*, *AggregateTacticalLevel.sms*, and the legacy *AggregateLevel.sms*).

27.1. The Aggregate Warfare Model 528
27.1.1 Unit Combat 528
27.1.2 Unit Footprints 529
27.1.3 Unit Posture 530
27.1.4 Unit Movement 531
27.1.5 Limited Munition Attack 532
27.1.6 Contamination Areas 532
27.1.7 Unit Sensors 533
27.2. Combat Engineering Objects 533
27.2.1 Concealed Obstacles 535
27.2.2 Creating Combat Engineering Objects 535
27.2.3 How Engineering Units Create Engineering Objects 536
27.3. Logistics 537
27.3.1 Receiving Supplies 538
27.3.2 Providing Supplies 539
27.4. Electronic Warfare 539
27.4.1 Jamming Communications 539
27.4.2 Jamming Radar 540
27.4.3 Sensing Electronic Emissions 541
27.5. Air Combat 541
27.5.1 Loadouts 542
27.6. Air Missions 542
27.7. Fuel Use and Aerial Refueling 544
27.7.1 Aerial Refueling Systems 544
27.7.2 Refueling Missions 544
27.8. Air Bases and Air Missions 545

27.8.1 Preparing Missions 546
27.8.2 Landing Air Units 547
27.8.3 Repairing Aircraft 547
27.8.4 Quick Reaction Alert Status 548

27.1. The Aggregate Warfare Model

VR-Forces supports a warfare model for aggregate-level scenarios. The aggregate warfare model requires use of *AggregateLevel.sms* or a similar SMS and only works with HLA Evolved and the MAK FOM extensions.

The aggregate warfare model is data-driven. Simulation objects have combat power and combat vulnerability. Their overall state is expressed as their health. These are abstract values that represent relative strengths of different units.

When opposing force simulation objects come into contact, they inflict damage on each other until one of the simulation objects is destroyed or breaks off combat. A unit is considered killed when its health is 0.

Some of the features that distinguish aggregate-level scenarios from entity-level scenarios include:

- Footprint. The area occupied by the unit.
- Posture. The unit's mode of interaction with the environment.
- Logistics. Management of personnel and matériel.
- Combat engineering objects. Tactical graphics that affect simulation object movement and health.
- Reports. Some data tracked in a simulation, such as personnel levels and pacing and tracking, is not used by the sim engine. It is sent over the network to be used by command and control systems and human participants.

Aggregate-level scenarios can include individual entities, such as surface vessels and aircraft. However, those entities also use the aggregate warfare model. They do not behave like they would if you were using the same entity types in entity-level modeling.

27.1.1 Unit Combat

Combat power is the ability of a simulation object to cause casualties and damage, and is a function of how many weapons it has, the size (destructive power) of their munitions, the rate of fire of the weapons, the accuracy of the weapons, and the effectiveness of the subordinate simulation objects in conducting an attack (tactics, coordination, target selection, and so on).

A simulation object's ability to engage in combat is expressed as a strength value in one or more of these domains:

- Anti-air
- Anti-tank
- High explosive
- Anti-personnel
- Anti-ship

The strength value describes the attrition rate per second imposed on the target by the attacking unit.

A combat domain has a range. When an opposing force simulation object is within the range of a combat domain for which a simulation object has strength, it attacks the simulation object.

Health is a function of the amount of equipment, personnel, or both in the simulation object, and the difficulty in destroying them. For example, a unit with 10 tanks would have twice the health of a unit with 5 of the same kind of tank. However, note that the initial health for a unit is set by the SMS designer in the Simulation Object Editor. It is independent of other parameters or resource variables. It is not computed based on the equipment in the unit. Also, health values in different domains are not usually comparable to each other. They are only meaningful relative to units in the same domain.

Simulation objects also have a vulnerability in each of these domains. Base vulnerability describes how susceptible a simulation object is to a given category of weapon. In particular, it describes how easy it is to hit and kill the equipment in the simulation object with the given category of weapon. For example, infantry is very susceptible to high explosive munitions, light armor fairly susceptible, and heavy armor not very susceptible.

Vulnerability is expressed as a modifier of the combat power (attrition rate) of the attacker. It is multiplied by the attrition rate to calculate the actual damage. For example, a Mech Inf CO US has an Anti-personnel strength of 78. A Mech Inf CO RU has an Anti-personnel vulnerability of 0.1. If the US company attacks the Russian company, the actual damage for personnel will be 78 * 0.1 = 7.8 per second.

Both combat power and vulnerability are modified by a variety of factors, including posture, MOPP level, sector, and morale. The combination of all modifiers results in the final attrition rate.

The attrition across all domains is applied to the simulation object's health. If health declines to 0, the simulation object is killed.

27.1.2 Unit Footprints

A unit footprint is an abstract representation of the area the unit occupies with vehicles, personnel, and equipment. A unit takes damage from events that occur within its physical footprint, such as explosions or other simulation objects firing into that area.

The physical footprint has four sectors: front, rear, left flank, and right flank. The interactions of the unit with other simulation objects in the scenario varies based on which sector another entity is encountered on. For example, a unit is typically much more vulnerable to attack on the rear and flank sectors.

Both the physical footprint size and the size of the sectors for a unit are configured with a base value, but can then change during the scenario depending on actions of the unit.

> The footprint is always circular. The footprint parameter describes the radius.

27.1.3 Unit Posture

Posture is an abstraction of the positioning of the equipment and personnel of a unit within its physical footprint, and its state of readiness for various activities.

The supported postures are:

- Deliberate-Defense. Planned, prepared. Lower mobility, lower vulnerability, moderate attack power. It is a positional defense.
- Hasty-Defense. Reactionary. Moderate mobility, moderate vulnerability, lower attack power. Happens when the unit first makes contact with the enemy. This is a positional defense, but with less time to prepare than deliberate-defense.
- Deliberate-Attack. Planned, prepared. Moderate mobility, moderate vulnerability, higher attack power. A fully planned attack executed as a deliberate advance using fire and maneuver techniques.
- Hasty-Attack. Reactionary. Higher mobility, higher vulnerability, lower attack power. Used for movement when enemy contact is expected.
- Reconnaissance. Mobile maneuver, locate and report on enemy, avoid detection, avoid engagement (if you can). Sensor signature and vulnerability are low, but so are combat power and speed.
- Travel. Very high mobility, but very vulnerable to attack and low attack power. Useful when the unit is moving from one location to another in a safe area.
- Rout. Health is less than 50% and unit is under attack. It cannot change to another posture until its health exceeds 50%.

A unit's posture affects:

- Footprint. The amount of space that the unit occupies.
- Sensing. For example, in reconnaissance posture the unit detects objects more quickly and at a greater distance than in a defensive posture.
- Sensor signature. The ability of other units to detect a unit is affected.
- Movement speed.
- Sector sizes.
- Combat power. Many weapon systems can only be used in certain postures. In particular, most cannot be used in Travel posture (which is the posture a simulation object is in when it is created).
- Vulnerability.

When a unit transitions between postures, it does so over time. Transitions may take up to several hours. The transition delay models the planning and preparation necessary to perform the task expected in the new posture. For example, deliberate defense is a well prepared defense that assumes planning fires, digging fighting positions, and so on. It significantly reduces the vulnerability of the defender. However, it may require 3 hours to

transition to this posture. Hasty attack is the only posture that does not require a long transition time, as it is the posture for quick actions on contact, meeting engagements, pursuits, and so on. Transition times between each posture are configured in the Simulation Object Editor.

Because transitioning between postures can take hours of simulation time, when you set up a scenario you must decide if you want units to start in a given posture or transition to it as part of the simulation.

Defense postures assume stationary defense.

Aircraft and ships generally do not distinguish postures, except that they are not battle-ready in Travel posture.

27.1.4 Unit Movement

Units try to move at their ordered speed. The movement speed of a unit over the terrain depends on many factors, including:

- The slope of the terrain and the broad categories of the terrain underneath the unit, alter the maximum possible travel speed.
- Roads allow units to move more quickly, while restricted areas like forests or cities slow them down.
- Rivers and water bodies may stop a unit completely, although bridges allow them to cross.
- Overlapping footprints cause units to slow down.

> The movement speed modifiers for slope, terrain, footprint overlap, MOPP, posture, and so on only affect maximum speed. If ordered speed is less than the net modified maximum speed, you will not see a change in speed due to these factors.

For movement speed calculations, the center point of the unit is used to determine what type of terrain the unit is on.

If the footprints of two units overlap, their movement is slowed based on their Max Speed Modifier When Overlapping Another Unit parameter. This parameter is set on the Movement tab for a unit in the Simulation Object Editor. The lower the value, the greater the movement penalty. When unit footprints overlap, the lowest modifier among all overlapping units is applied to all of them.

> Aggregate-level scenarios do not support collision avoidance among simulation objects.

The effect of terrain features on movement is configured in a unit's movement system. For details, see "Configuring Aggregate-Level Movement Restrictions" in *Adding Content to MAK ONE Applications Reference Manual*.

VR-Forces supports a variety of combat engineering objects that affect entity movement. They may cause a unit to go more slowly or they may damage the unit. For information about combat engineering objects, see "27.2. Combat Engineering Objects" on page 533.

27.1.5 Limited Munition Attack

Most aggregate-level combat is automated and continues until a combatant is killed or combatants run out of ammunition. There are no specific fire or detonation interactions involved and combat essentially takes place in the background. The aggregate warfare model also supports limited munition attacks, in which a limited amount of munitions are used and the attack then ends. In the case of artillery, this is a specified number of rounds from a battery, delivered over a short time. In the case of large munitions such as missiles and bombs, the attack for each munition is generally instantaneous. VR-Forces does not model the time it takes for a munition to reach its target.

> Not all missile attacks are modeled as limited munition attacks. Anti-tank missiles in ground units, for example, are launched by multiple launchers in the unit against multiple targets in the defender, and are modeled with normal combat power.

Individual missile attacks typically make use of a hit factor from the attacker and a defense factor from the defender to determine a hit probability. The missile must hit before its combat power is applied in an attack.

27.1.6 Contamination Areas

The aggregate warfare model supports nuclear, biological, and chemical contamination areas (NBC areas) that may damage or impede a unit. You can create contamination areas as tactical graphics, or simulation objects with nuclear, biological, or chemical weapon systems can create them as a result of attacking with these weapons.

Contamination areas have parameters that are similar to other areal tactical graphics, plus these parameters:

- Agent type. The type of contamination. It determines the configuration used for damage calculations.
- Terrain type. Informational only; not used by VR-Forces.
- Topography type. Informational only; not used by VR-Forces.
- Vegetation type. Informational only; not used by VR-Forces.

The terrain, topography, and vegetation types are published by VR-Forces, but do not affect unit modeling or interaction. They may be useful to other applications that interact with VR-Forces.

By default, NBC hazard areas are placed on the Contamination Areas overlay.

Contamination (NBC) Areas and MOPP Level

When a unit is attacked by NBC weapons or enters an NBC area, it suffers attrition. It stops its current task and increases its mission oriented protective posture (MOPP) level until it stops suffering attrition. Then it resumes its task. However, any contamination that took place persists.

If a unit is moving and it detects an NBC hazard area in its path, it stops moving and increases its MOPP level to a point that will allow it to pass through the hazard area without suffering attrition.

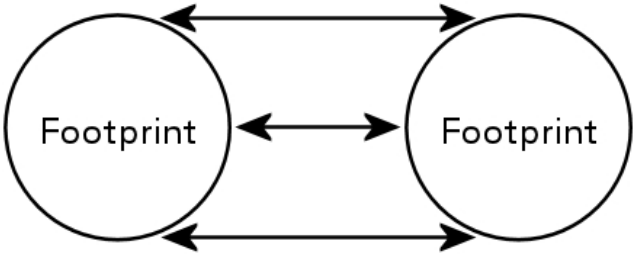
Increasing MOPP level affects a unit's speed, combat effectiveness, sensor sensitivity, posture transition time, and vulnerability.

Some types of contamination are benign enough that they cause little, if any, damage to simulation objects. Other types, such as gamma rays, cause attrition even at the highest MOPP level.

27.1.7 Unit Sensors

Simulation objects in aggregate-level scenarios use the same sensor signature mechanism as simulation objects in entity-level scenarios. Sensor systems allow them to detect other simulation objects in various domains. Sensor signatures indicate how detectable they are by other simulation objects in the applicable domains. The main difference between simulation objects in the two types of modeling is in the detection checks. Simulation objects in entity-level scenarios do a line-of-sight check from the center of the simulation object. Simulation objects in aggregate-level scenarios do three checks: one from the center of the outer ring of the footprint and two from the edges of the circumference of the footprint, as illustrated in Figure 210.

Figure 210. Aggregate-level sensing



27.2. Combat Engineering Objects

VR-Forces supports a variety of combat engineering objects that affect unit movement. They may cause a unit to go more slowly or they may damage the unit. Combat engineering objects (CEOs), such as fortifications, can also protect a unit, in that they affect the effectiveness of opposing forces.

CEOs are displayed as tactical graphics. You can create combat engineering objects from the Create Object Palette. Also, units can create combat engineering objects in response to combat engineering object tasks.

Units can counter the presence of combat engineering objects by breaching them, bridging them, defusing them, or destroying them.

Combat engineering objects can affect units:

- Mobility. CEOs can decrease entity mobility. Mobility modifiers are configured by mobility type. For instance, an object might affect the mobility of soldiers on foot more than tanks. When a simulation object comes in contact with an engineering object, it checks the object type's mobility modifier against the entity's mobility

type. The entity slows down based on the modifier value.

The modifier is scaled based on the completion percentage of the object. Modifiers only affect the maximum speed, so speed is affected only if the modified maximum speed is less than the current speed of the unit.

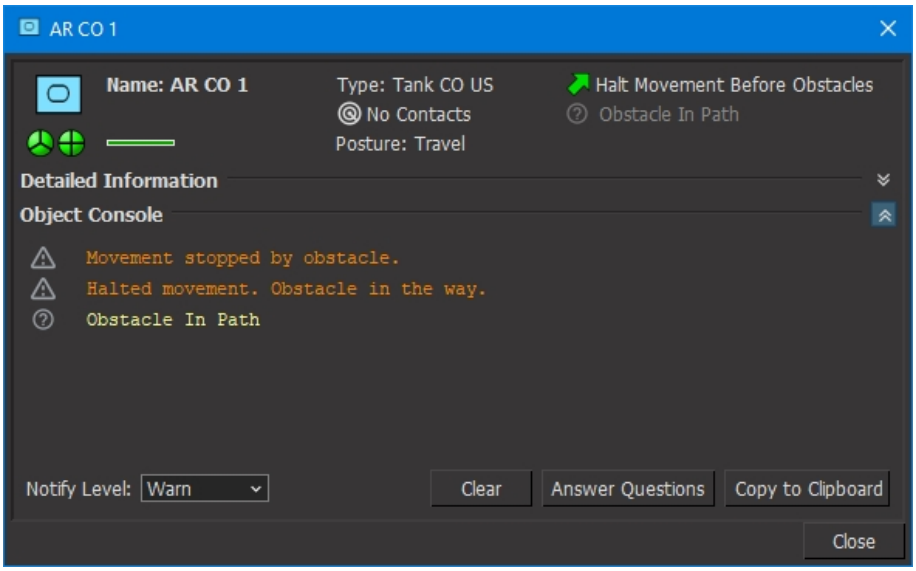
If a unit encounters an obstacle that obstructs its movement or may cause damage, it stops moving and sends a message to the console indicating that it is obstructed (Figure 211). The user must then tell it what to do. The console message may use the question and answer capability. (For information about console questions, see "21.9.3 Answering Questions from Scripted Tasks" on page 485.)

> The unit may not be able to move until you respond to the question.
>
> If an obstacle is concealed, the unit does not stop. For information about concealed obstacles, see "27.2.1 Concealed Obstacles" on the facing page.

- Mobility enabling. Certain types of engineering objects are treated as terrain features by a moving unit. For instance, a simulation object may determine that an engineering object represents a road or bridge, giving it additional path planning options. This capability is either on or off.
- Vulnerability. Vulnerability modifiers are configured by weapon type. For instance, a fortification might do very well at stopping anti-personnel weapons, but may do little or nothing to stop high explosives. When a simulation object under fire contacts an engineering object, it checks the object type's vulnerability modifier against the attacking weapon's category. This may reduce the amount of damage the entity takes from that weapon.

  The modifier is scaled based on the completion percentage of the object.

Figure 211. Question in entity console



- Visibility. When a simulation object contacts an engineering object with a visibility modifier, its visibility to opposing simulation objects is increased or decreased accordingly.

  The modifier is scaled based on the completion percentage of the object.

- Attrition over time. When a simulation object contacts an engineering object that causes damage (for example a minefield), that object continues taking damage over time as long as it remains in contact. The damage is configured as one of the existing damage/weapon categories, for example, high-explosive. Simulation objects with more or less vulnerability to that damage type take more or less damage accordingly.

  The attrition is scaled based on the completion percentage of the object.

- Immediate attrition. Some engineering objects cause one-time, immediate damage to any unit that comes in contact with them. For instance, a simulation object might trigger an IED. Usually the engineering object is destroyed in the process.

  The modifier is scaled based on the completion percentage of the object.

  The various modifiers are configured for each CEO in the Simulation Object Editor.

27.2.1 Concealed Obstacles

By default, minefields, booby traps and unexploded ordnance objects are concealed. (This attribute is configurable in the Simulation Object Editor, so you can set it for other objects if you want to.) The engineering object sensor does not automatically detect concealed objects within its sensing radius. The sensor has a random (configurable) chance of detecting the object each sensor cycle that the object is within range.

If a sensor detects a concealed minefield, the Mark Minefield reactive task marks it as detected. Thereafter, the minefield is always 100% detectable. (The Mark Minefield reactive task is enabled by default.)

Unit movement systems check the concealed flag for obstacles. For non-friendly obstacles that are concealed, the "stop and ask" behavior is suppressed unless the object has been specifically detected by the sensor. Therefore, if the sensor has not detected the object, the unit enters it without first stopping and alerting you to the presence of the object.

Concealed objects are always discovered if they are causing damage to a simulation object.

27.2.2 Creating Combat Engineering Objects

You can create combat engineering objects directly from the Create Object Palette or you can task simulation objects with the appropriate systems, such as combat engineering units, to create them.

Combat engineering objects are implemented as points, lines, and areas, so you can create them the same way that you would create tactical graphics. Most areal CEOs are fixed size, which is configurable in the Simulation Object Editor. You just have to specify the lower left corner of the area.

When you create a CEO from the Create Object Palette, you can give it a name, specify a layer and set all the parameters you would expect for that type of object. When you task a unit to create a CEO, it is given a default name and you can only specify its location.

Many tactical graphics have the same default names as CEOs, for example, fortified lines and minefields. They may also look the same on the map, however, tactical graphics do not affect unit movement and combat the way that CEOs do. To help distinguish tactical graphics from CEOs, CEOs created by units are placed on the Combat Engineering Object layer. It is recommended that if you create CEOs from the Create Object Palette, that you place them on the CEO layer.

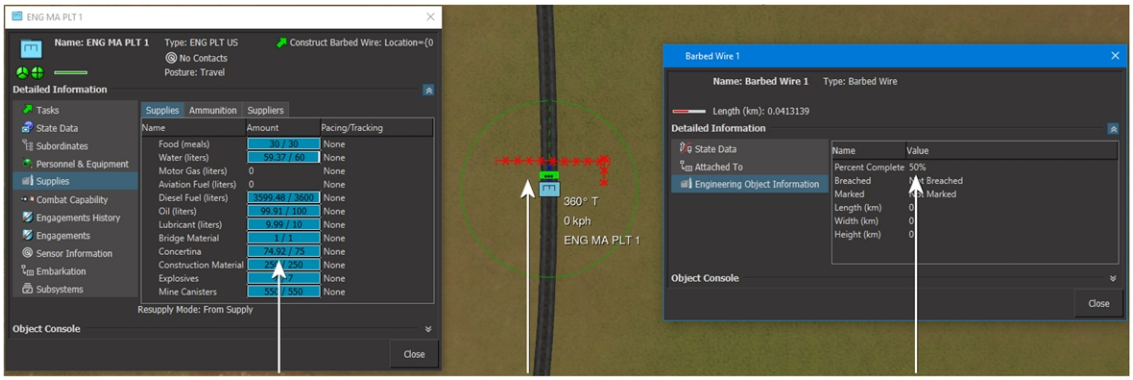
27.2.3 How Engineering Units Create Engineering Objects

When a unit with engineering capabilities creates a CEO, it expends construction material at a rate specified in the object's model (configured in the Simulation Object Editor). Depending on the type of objects, the process can take hours of simulation time.

While an object is partially complete, its effectiveness is reduced. Some effects, such as enabling mobility over previously impassable terrain (for example by using bridges), do not take effect until the object is complete.

> You can follow the progress of object construction in two ways. You can view the engineering unit's use of construction material on the Supplies page of its Information dialog box. You can view the percent of the object's completion on the Engineering Object Information page of the Information dialog box for the object (Figure 212).

Figure 212. Barbed wire under construction



Units create combat engineering objects as follows:

1. The unit moves within range of the location to create the object.
2. The unit creates a new object that is 0% complete. The object is displayed on the terrain.
3. The unit begins constructing the object. As it does so, its supply of construction material decreases.
4. When the object is complete or the unit runs out of construction material, it ends the task.

If a creating unit is interrupted and must halt construction, it notifies that object that construction has stopped. The object stays at its current level of completion, which may or may not be useful, depending on the object type. A constructing unit might also reduce its rate of construction if it takes damage.

If the constructing unit is able to resume construction or another unit is tasked to improve the object, it moves within range of the object and construction starts again.

Multiple units with construction capability can work together. To task a second unit to assist the first, task it to improve the object under construction.

You can also set the completion percentage of a combat engineering object with the Percent Complete set data request.

27.3. Logistics

Simulation objects have resources, such as fuel, ammunition, and food. Over time, and particularly during combat, these resources are consumed. Logistics is the process by which a unit resupplies itself.

Units can resupply these resources:

- Health
- Food
- Water
- Motor gas
- Aviation fuel
- Diesel fuel
- Oil
- Lubricant
- Anti-Tank
- High-Explosive
- Anti-Personnel
- Anti-Air
- Anti-Ship
- Large-Munition
- Other

Anti-Tank, High-Explosive, Anti-Personnel, Anti-Air, and Anti-Ship refer to ammunition in that category. There is no distinction made between different ammunition names within a category. Large-munition covers all other categories of ammunition, such as indirect fire munitions, bombs, and missiles. Other supplies covers any other type of supply, including engineering materials.

The Supplies page on the Information dialog box indicates when a simulation object is receiving or providing supplies. You can display supply lines that show when a supply unit is supplying another unit (Figure 213). In Figure 214, green arrows show the supplies that the entity is receiving.

Figure 213. Supply information and supply lines

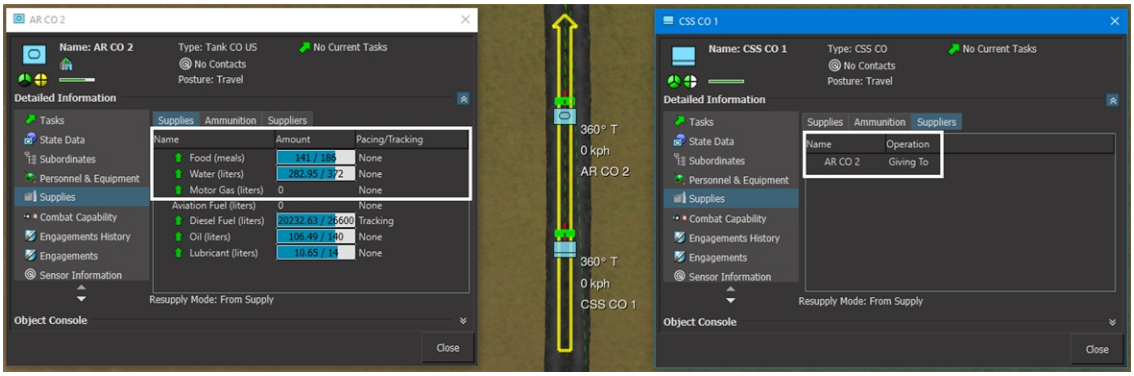
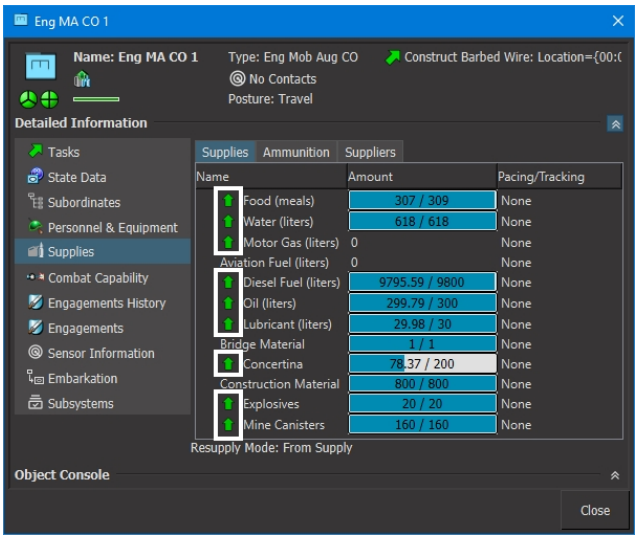


Figure 214. Supplies being received



When a simulation object is engaged in combat, resupply pauses for a time.

27.3.1 Receiving Supplies

To receive supplies, units can:

- Take supplies from a supply unit. Units take supplies from a friendly supply unit that is within range and has notified them that it can provide supplies. The resupply range is specified in the Simulation Object Editor as part of a supply system.
- Auto Resupply Continuously. Supplies increase continuously at a configurable rate.
- Auto Resupply Periodically. Simulation objects refill all supplies at specified intervals. The interval is configurable.
- Refueling. Air units can refuel from tanker units. For details, see "27.7. Fuel Use and Aerial Refueling" on page 544.

The ability to resupply can be restricted if a simulation object is moving, attacking, or defending. This is configured in the Simulation Object Editor on the Supplies tab. You can change how a simulation object receives supplies using the Set Resupply Mode task. If they are configured for airborne refueling (Can Receive Fuel While Airborne parameter), airplanes can receive fuel while they are flying, but must be on the ground to receive any other type of supply.

27.3.2 Providing Supplies

Simulation objects can provide supplies to other simulation objects if they have a resupply system (*resupplier.sysdef*). A supply unit can be configured to provide any of the supplies that are supported. A supply unit can provide supplies to any number of other simulation objects. It is assumed to have its own supply source, so it never runs out. You can change a supply unit's status with the Change Supplying Command task.

A supply unit runs a background process that tests for friendly simulation objects in its area and notifies them that it is available to provide supplies. It checks for conditions that might prevent a simulation object from receiving supplies. It stops providing supplies to simulation objects that move out of range.

Aircraft units that have a refueling capacity can refuel other air units that request refueling.

27.4. Electronic Warfare

The aggregate warfare model supports three types of electronic warfare: communications jamming, radar jamming, and sensing electronic emissions.

27.4.1 Jamming Communications

Each unit has an EW-Comms-Dependence and an EW-Defense-Strength property. The dependence property indicates how much it can be affected if its communications are jammed. The defense factor indicates how well its technology and practices protect it from communications jamming. Units also have, in their state properties, a list of jamming attacks being made against them (type and strength) and a net EW-Comms-Degradation-Percentage.

When a simulation object attacks another entity with jamming, it sends EW attack information to the target.

> Simulation objects jam all enemy simulation objects in range; there is no target selection. Friendly simulation objects are not affected.

When a simulation object suffers communications jamming attacks, the strengths of all attacks are added together and compared to the defensive strength. The ratio determines a modifier between 0 and 1, which is multiplied by the EW-Comms-Dependence. The result is a degradation percentage. This degradation percentage is applied to reduce combat power, and increase vulnerability and posture change time (by a maximum factor of 2.0).

Units can have a system called an aggregate-comms-jammer. This system has parameters for strength, maximum range, and a set of allowed postures. Strength falls off with $1/r^2$ (r is range in kilometers). This system can trigger EW attacks using the Make EW Attack background scripted task. It has a jammer controller that can publish an emitter.

You can enable or disable the emitter using the Emitter set data request. Turning it on activates jamming, and turning it off deactivates jamming. If the emitter is turned on while the simulation object is not in an allowed posture for jamming, a warning is printed to the console. In this case the emitter is on, but it is not used for jamming attacks.

When jamming is enabled the Make EW Attack script determines which jamming systems on the simulation object are on, and then uses them to attack all targets within range. It updates (new) EW-Attacks and Jamming-Strength-On state properties on the simulation object. The Jamming-Strength-On property summarizes the strength from all jammers the simulation object has turned on, and is used for emissions sensing. The Update_EW_ Degradation background script computes the effects of EW attacks on a simulation object.

The MI PLT (military intelligence platoon) unit is configured to have an aggregate-comms-jammer.

27.4.2 Jamming Radar

Each unit has a state property Radar-Jamming-Strength-Receiving, which is updated by the Update_EW_Degradation background script. This is the total radar jamming strength being received.

Radar jamming can:

- Reduce the sensitivity of radar sensors so they cannot detect targets.
- Reduce the hit factor of radar-homing missiles so they have a smaller hit probability.

The active-radar-sensor system has a parameter called **jamming-defense-factor**. This factor reflects the technology of the radar (not power) and is used with the jamming strength received to compute a sensitivity modifier for the sensor. The modifier varies from 0 to 1.

The weapon systems that use a controller that uses an aggregateReleaseBombControllerDescriptor have parameters **can-be-radar-jammed** and **jamming-defense-factor**. This includes these weapon systems:

- aggregate-antiair-missile
- aggregate-antiship-missile
- aggregate-land-attack-missile
- aggregate-release-bomb
- aggregate-unguided-bomb

If **can-be-radar-jammed** is true, then **jamming-defense-factor** is used with jamming strength received to calculate a modifier for the hit factor of the weapon.

Weapon systems are configured with either a constant or variable for **can-be-radar-jammed**, to provide examples of different guidance systems:

- Anti-air missiles use a variable, to allow modeling of IR or radar-homing missiles.
- Anti-ship missiles set **can-be-radar-jammed** true, thus assuming radar guidance.
- Land attack missiles use a variable.
- Release-bomb omits the parameter (default is false), to simulate optical guidance.
- Unguided bomb omits the parameter (default is false).

The aggregate-radar-jammer system can publish an emitter and can be turned on and off using Emitter set data request. It enables the EW_Attack script.

The radar jammer strength falls off with $1/r^2$ (r is range in kilometers).

The EA-18G entity is configured with a radar jamming system.

27.4.3 Sensing Electronic Emissions

The emission sensor domain allows simulation objects to sense RF electromagnetic emissions from simulation objects. Each simulation object has an emissions signature, which is a parameter for each object type. This can be detected by the SIGINT sensor. This is the "normal background signature".

There is a sensor signature modifier for emissions. VR-Forces determines if specific emitters are turned on, thus adding to the normal background signature for the simulation object .

The emissions signature modifier has a parameter, **jam-strength-factor**, that converts jamming strength to a signature value. It is set in the platform files.

There is a sensor system emissions-sensor for emissions. It is called a SIGINT (signals intelligence) sensor. The MI PLT unit is configured with a SIGINT Sensor.

27.5. Air Combat

All air-to-air and air-to-ground engagements are executed using the limited munitions attack capability in the aggregate model. The core of this model is that a single munition is fired at a time, and then a hit calculation is performed. The hit calculation uses the hit-factor of the shooter and compares it to the defense factor of the target to determine a hit chance. A random draw is done and compared to the hit chance to determine if an actual hit takes place. If the hit occurs, then the attack power of the munition is applied to the target.

Aircraft units can represent one or more aircraft. Each aircraft has a primary equipment attribute, which is usually the airframe. When aircraft units engage in combat, damage is calculated against the primary equipment. For example, if an F-16 aircraft has the F-16 airframe as its primary equipment, a unit that represents four F-16s has a primary equipment count of four. When that unit takes damage, no more than one F-16 can be damaged by each hit.

Each aircraft type has a loadout that is defined in the Simulation Object Editor. A loadout specifies the equipment and weapons for the aircraft. At runtime, you can set the loadout for an aircraft unit. The loadout for a unit determines how effective it is when it attacks another simulation object.

The results of air-to-air engagements are determined by these rules:

- Sensor capabilities. The sensor range and sensitivity, and the sensor signature of the target determine the maximum range for detection, and therefore the maximum range for engagement of a target.
- Weapon type. The weapon types on the aircraft determine the maximum engagement range when engaging with a weapon. The weapon type also determines the hit factor that affects the chance of a weapon hitting, and the total damage done if the weapon does hit.
- Tactical Data Link. If the aircraft unit is equipped with a Tactical Data Link combat system, the unit receives a bonus to its defense factor.
- Number of aircraft. Defending aircraft units receive a combat maneuvers bonus to their defense factor. Attacking aircraft have a smaller attack interval when the number of aircraft is larger, causing them to fire more missiles in a shorter period of time. This bonus is based on the current number of aircraft in the unit. It is reduced if some of the aircraft are destroyed during combat.
- Battle management support. If the attacking unit has active spot-report data on the target aircraft from other units in the scenario, it receives a hit-factor bonus when engaging the target. A defending unit also receives a defense-factor bonus if it has active spot reports on the attacking entity.

The details of an engagement are displayed in the Information dialog box.

The AWACS unit is configured with long range radar. It can provide spot report tactical data to other aircraft units, which allows them to engage units that are beyond the sensor range of their aircraft.

27.5.1 Loadouts

Aircraft loadouts are defined in the Simulation Object Editor for each aircraft type, in the Combat tab. Each defined loadout specifies the quantity of weapons and equipment in the loadout. One loadout can be marked as the default. At scenario runtime, you can set the loadout for an aircraft unit and the unit will be equipped with the predefined loadout items.

The weapons and equipment determine the effectiveness of the attacks a unit can make. Additional fuel tanks can increase the range of aircraft. Jammer pods increase defense against radar guided weapons.

27.6. Air Missions

Missions are abstract command objects that link together aircraft that will be launched as a flight. A mission has these attributes:

- Name (call sign).
- List of base aircraft in mission.
- Aircraft loadout.
- Target launch time.

- Time to start preparing for launch. This is computed from the target launch time and an estimate of the aircraft preparation time.
- Simulation object type for the air unit that will be created for this flight.

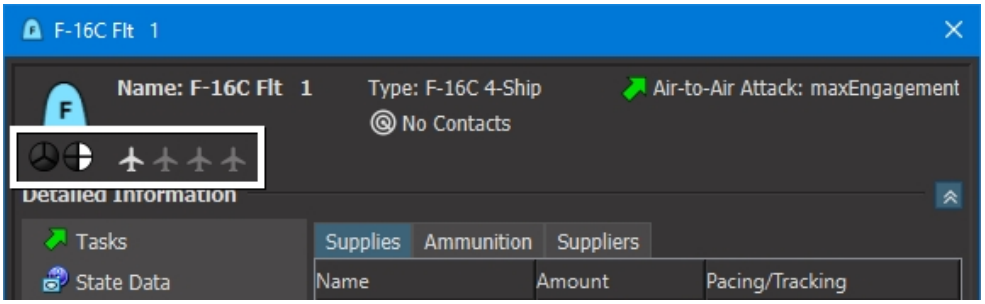
Air missions are carried out by aircraft units. A mission can be carried out by a single unit (which can represent multiple aircraft) or by multiple units. A multiple unit mission consists of multiple single units and a single "Mission Superior" object that is the superior of all the single units that are part of the mission.

The icons for air units have graphics that indicate the number of aircraft in the unit (Figure 215). Information dialog boxes also show the number of aircraft. If a unit suffers attrition, the number of graphics is reduced (Figure 216).

Figure 215. Aircraft unit icon



Figure 216. Aggregate information after attrition



The units that execute a mission can be created as part of the initial scenario laydown, they can be created during runtime by plans, players, or instructors, or they can be launched by an air base.

For single unit missions, the mission orders are issued directly to the unit. For multi-unit missions, the mission orders are issued to the mission superior.

Mission orders may include a task to land at an air base at the end of the mission. The units executing the mission are deleted when they land, so the mission ceases to exist. Any report of mission status should be made by the mission plan before this occurs.

VR-Forces does not have any specific mission management features. Users can manage missions with the global and unit plans, through player interaction, or through externally developed applications that use the remote control API.

27.7. Fuel Use and Aerial Refueling

Fuel is a supply that is configured in the SOE (on the Supplies tab) and consumed based on aircraft activity. Air units use fuel based on their altitude and speed. When an aircraft unit is in a holding pattern, fuel use is calculated based on the default speed. VR-Forces interpolates values for altitudes and speeds that fall between the configured values.

For tankers, the fuel available for refueling other aircraft is assumed to be the same fuel that the tanker itself uses to fly. When refueling is complete, aviation fuel supplies are transferred from the tanker to the receiving aircraft.

27.7.1 Aerial Refueling Systems

An aircraft unit can refuel other aircraft if it is configured with an aerial refueling system. The fueling system can be a flying boom type, a drogue-and-probe type, or a combination. The configuration cannot be changed during a simulation run.

The boom system specifies a maximum fuel transfer rate. It can also be configured to have drogue lines on it, with a different transfer rate for the drogues.

Drogue systems define the number of drogues and the fuel flow rate for each fuel line.

Aircraft that are to receive fuel must be configured with an aerial refueling receiver system. The aerial refueling receiving system specifies whether the system is a boom or drogue system and what the maximum fuel transfer rate is. Air units may only refuel from tankers that have compatible refueling equipment.

Refueling systems and fuel receiving systems are assigned on the Additional Systems tab in the SOE. *AggregateLevel.sms* has two tanker units, a KC-135A with a boom system, and a KC-130J with two drogues.

27.7.2 Refueling Missions

Air units can be tasked to refuel from a tanker unit with the Refuel from Tanker task. Tanker units provide fuel in response to a Refueling Mission task. When a tanker unit receives a Refueling Mission task, it flies to the designated location, flies a race track pattern, and services aircraft that need fuel. It can continue to refuel aircraft until its fuel level falls below a configurable threshold. (For details about the Refueling Mission task, see "35.8.6 Refueling Mission" on page 739. For details about the Refuel from Tanker task, see "35.8.5 Refuel from Tanker" on page 738.)

The tanker coordinates with the refueling aircraft by exchanging text radio messages. The aircraft unit initiates a refueling request and says what type of system it is using and how much fuel is required. The tanker then either rejects the request (based on refueling system type and amount of fuel available) or accepts it. The aircraft unit is expected to move close to the tanker and report in with a message. When the tanker has a refuel line available, it tells the unit to move to the tanker and begins fueling it. The transfer rate is dependent on the transfer rates of the tanker and the aircraft, and the number of aircraft and drogue lines available. Supply transfer messages are sent to the refueling aircraft periodically, and corresponding aviation fuel resources in the tanker are reduced.

When refueling is complete, the tanker notifies the refueling unit.

If multiple aircraft units request refueling, the tanker coordinates with each of them and refuels them each in turn.

27.8. Air Bases and Air Missions

An air base is a special type of entity that is placed on the ground. Air bases have a variety of resources that they can use to prepare, launch, and recover air missions. They do this using a background process (Manage Air Base) and several tasks and set data requests.

An air base represents these physical features:

- Apron. The aircraft that are assigned to an air base wait on the apron to be assigned to missions, prepared for launch, or sent to a hangar to be repaired. The apron has a maximum capacity of aircraft.
- Hangar. When a damaged aircraft is actively undergoing repair, it is moved to a hangar. Hangars have a maximum capacity of aircraft.
- Runway. Aircraft that are prepared for a mission taxi on the runway area for a few minutes before taking off.
- Operational Readiness Platform (ORP). The location at the end of a runway where aircraft on Quick Reaction Alert (QRA) are stationed. The ORP has a maximum capacity of aircraft.

Air bases have these resources:

- Aircraft. All of the aircraft at an air base are on the ground. Air bases do not track the aircraft that are executing missions. When an aircraft takes off, it is deleted from the air base records. Aircraft have these attributes:
  - Type (equipment name).
  - Health. Health is expressed as a percentage and represents the status of the air frame and the amount of fuel on board.
  - Number of munitions loaded.
  - Location on the air base.

  An aircraft can be in one of the following states:
  - Available. The aircraft is at 100% health and is available to be assigned to a mission.
  - Assigned. The aircraft has been assigned to a mission. Some time before the mission launches, an assigned aircraft begins fueling and arming and its air crew prepares.
  - In Repair. The aircraft is in this state if its health is less than 100%. It can only be actively repaired (have its health improved) if it is in a hangar and has ground crew assigned to it.
  - QRA. The aircraft has been assigned and prepared, and is waiting at the ORP.
- Air crews. Each aircraft must be manned by one air crew. An air crew can man any type of aircraft. Air crews can be available or assigned. Air crews are assigned (to missions) when the air base is tasked to launch aircraft.

- Ground crew personnel. Ground crew personnel are used to repair, fuel, and arm aircraft. They may be in one of the following states and locations:
  - Available. Not assigned to any activity.
  - In-Hangar. Repairing aircraft.
  - On-Apron. Fueling and arming aircraft.
- Fuel. A single type of fuel is used to fill all aircraft prior to launching them on a mission.
- Munitions. Large munitions (bombs, missiles, and rockets) are aggregated into one count of munitions for the air base. When an aircraft is assigned a mission and given a loadout, the quantity of munitions in the loadout is deducted from the base munition count.
- Runways. An air base can have one or more runways. Runways have a length and a heading. The length of a runway determines whether aircraft can land at the air base.

You can specify the resources of an air base with these set data requests:

- "41.4. Assign Aircraft" on page 916
- "41.17. Facilities" on page 919
- "41.32. Personnel and Supplies" on page 923

27.8.1 Preparing Missions

When an air base has pending missions, the Manage Air Base process handles the preparation of the aircraft as follows:

1. It estimates the preparation time and sets a time to begin preparation so that the flight can launch at the desired time.
2. When the preparation time arrives, the process assigns ground crew to the aircraft and computes the time at which fueling, arming, and preparing the air crew will complete. These tasks occur in parallel.
   - Fueling time requires a setup time and depends on how much fuel is taken on board.
   - Arming time is constant. It is independent of the loadout.
   - Crew preparation time is constant. It is independent of the type of aircraft.
3. When all preparation tasks are complete, the ground crew is made available to the air base.
4. When all aircraft in the flight are ready, they taxi onto the runway.
5. An air unit is created and the aircraft counts in the air base are decremented to remove the flight from the air base.

The *AirBase.ope* platform file includes parameters for the Manage Air Base script that determine:

- The size of the ground crew needed to fuel an aircraft.
- The size of the ground crew needed to arm an aircraft.
- The time required to arm an aircraft.
- The time required to set up refueling.
- The rate at which fuel can be transferred to the aircraft.
- The time required for the air crew to prepare.
- The time required for the flight to taxi and take off.

27.8.2 Landing Air Units

When an air unit wants to land at an air base (Land at Air Base task), it sends a message to the air base requesting to land. It tells the air base the number of aircraft in the unit and how long the runway needs to be for it to land. The Manage Air Base background process handles this request.

If the air base has room for the aircraft and its runway is long enough, it tells that air unit that it can land. The air unit flies to the air base and then requests final permission to land.

The air base determines the properties of the air unit:

- The aircraft type.
- Number of aircraft.
- Air frame health percent.
- Fuel in the tanks.
- Number of munitions.

When the air unit lands, the unit is removed from the scenario. The air base adds the aircraft to its list, and makes them either available or in-repair (if they are damaged). An air crew is added to the available air crews for each aircraft. Fuel and munitions are added to the air base's fuel and munition counts. The count of aircraft on the apron is incremented for each aircraft.

27.8.3 Repairing Aircraft

The Manage Air Base background process manages aircraft repair. When an air base has damaged aircraft, it moves them into the hangar and assigns ground crew personnel to them. The process can only start this repair if there is space in the hangar and ground crew available. Aircraft in the hangar are repaired at a rate that is proportional to the number of ground crew assigned. When the aircraft is at 100% health, the ground crew are made available again, and the aircraft is moved back to the apron and becomes available.

The *AirBase.ope* platform file includes parameters for the Manage Air Base script that determine:

- The repair rate (% repair / ground crew / second).
- The maximum number of ground crew that can be assigned to an aircraft.

27.8.4 Quick Reaction Alert Status

When aircraft are assigned Quick Reaction Alert (QRA) status, they prepare to take off and taxi to the Operational Readiness Platform (ORP), which is a set of parking spaces at the end of the runway. When events require it, one or more aircraft are ordered to take off and intercept a target. Aircraft and crews cannot remain prepared indefinitely, so after some time they must be returned to unprepared status at the base.

You can specify a level of quick reaction capability. For example, some QRA aircraft might keep their engines running, while others are fueled and armed on the apron with the crew in a ready room. In general the faster the reaction capability, the less time the aircraft can be on QRA.

Aircraft in QRA status remain part of the air base inventory rather than being formed into a unit.

You assign aircraft to the QRA status with the Prepare QRA task. You launch them with the Launch QRA Aircraft task. For details, see "35.4.3 Prepare QRA" on page 731 and "35.4.2 Launch QRA Aircraft" on page 731.



28. Movement in Aggregate-Level Scenarios

This chapter provides an overview of how simulation objects move in aggregate-level scenarios. Note that there is no formation movement; that exists only at the entity level.

28.1. Movement Systems and Terrain 550
28.2. Individual Unit Movement 550
28.2.1 Air 550
28.2.2 Land 550
28.2.3 Surface and Subsurface 551
28.3. Group Unit Movement 551
28.3.1 Air 551
28.3.2 Land 551
28.3.3 Surface and Subsurface 551

28.1. Movement Systems and Terrain

Aggregate-level movement systems use modifiers based on vector data in the terrain. These are area features from geospatial information, such as land use or municipal boundaries. For example, the areas might be forest, ocean, cropland, or municipality.

You can configure aggregate-level movement systems to associate movement modifiers with the vector features. The *./appData/settings/featureConfig.txt* file contains a set of expressions that define feature queries that the VR-Forces sim engine uses. The file includes comments that explain the syntax.

You can create feature additional configuration as needed. For any earth file, you can create an *.mfc* text file with the same name as the earth file. An *.mfc* file defines the feature and mobility queries that movement systems will use. For more information on how features affect movement, see "6.17. Specifying Terrain Categories" on page 221.

A system definition file (*.sysdef*) defines the impact of a mobility query on the speed of movement. For information on configuring movement systems, see "Chapter 79. Configuring Movement Systems" on page 1565.

28.2. Individual Unit Movement

This is information about how aggregate-level simulation objects simulate the movement of individual units. For information about the specific tasks, see "35.5. Movement Tasks for Aggregate-Level Scenarios" on page 732.

28.2.1 Air

For basic movement tasks such as Move To or Move Along, the aircraft snap to at least a minimum altitude and then transit to the destination. The details of fixed- and rotary-wing flight are abstracted, therefore there is no pitch nor roll in flight. Once an air unit reaches a destination, it may appear to be stationary. This is another part of the abstraction of the unit's movement. It simulates the unit operating in that general vicinity; the airframes are not floating.

There are other, more complicated tasks that build on basic air movement, such as Patrol Route, Launch Flight, Escort Air Unit, and so on.

28.2.2 Land

Land unit movement is based on move-to, which is designed to take a direct path to the destination. The unit remains on the ground and does not automatically avoid features. For example, if you task a unit to cross a lake, it stops at the shore unless it is equipped for amphibious movement.

Land units can path plan along roads using the Move to Location (Plan Along Roads) or Move to Waypoint (Plan Along Roads). The Move task has an option, Use Roads, which also causes the unit to move along roads where available.

During their movement, the land unit simulation object does not avoid any 3D models of buildings. Aggregate-level movement is abstract, moving through the region. The MAK Earth Aggregate (online) terrain does not extrude buildings from their footprints. Instead,

aggregate-level land-based movement systems rely on modifiers that are associated with municipal feature areas. In that case, it employs a concept of Urban area. There are also modifiers for forest, swamp, and so on.

There is a movement penalty when the footprints of land units overlap.

28.2.3 Surface and Subsurface

When tasked with direct movement, subsurface units move in a straight line to the specified destination. They may run aground.

When tasked to plan a path to a destination, subsurface units try to stay on waterways (oceans and rivers) and avoid land when calculating the route.

Subsurface units adjust depth based on bathymetry in the terrain database, if there is any. The subsurface unit remembers its depth at the time the task is assigned. In areas where the water is shallower, the subsurface unit moves closer to the surface. In areas where the water is deeper, the subsurface unit moves towards its remembered depth.

28.3. Group Unit Movement

This is information about how aggregate-level simulation objects simulate the movement of individual units.

28.3.1 Air

Air unit movement is direct. At the aggregate-level, destinations are computed for subordinates based on their relative location within the group while they perform the task.

28.3.2 Land

Groups of ground units can move directly to the destination or they can plan along roads.

Group planned movement along roads first generates a route for the group. Next, it the calculates destinations for the subordinates based on their relative positions at the time of tasking. After that, the subordinates queue along the route until they reach the end of the route. Finally, they unpack once they arrive at the destination.

28.3.3 Surface and Subsurface

Surface and subsurface units move directly to the specified destination. They do not plan along or follow coastlines, nor avoid shallows. They can run aground.

There is no detailed formation model for groups. Subordinate destinations are calculated based on relative position within the group at the time of tasking.



VI. Tasks

This section describes tasks and scripted tasks and sets.

Introduction to Tasks

VR-Forces simulation objects can execute a variety of movement and tactical tasks. These tasks can be assigned in plans, through the Task menu, or in the Command Panel. Tasks assigned outside of a plan are called independent tasks.

Some tasks are mutually exclusive with other tasks, for example, a simulation object cannot execute two different movement tasks at the same time. Other tasks, such as Send Text Message, can run concurrently with movement tasks. When you give a task to a simulation object, if the new task is mutually exclusive with the current task, the simulation object immediately stops the current task and begins the new one. If the new task is not mutually exclusive with the current task, the simulation object executes the new task and continues with the previous task. For information about how to configure mutually exclusive tasks, see "29.7. Configuring Task Execution Rules" on page 580 .

If a simulation object has a plan, when you give it an independent task, it abandons the rest of the plan.

Independent tasks are useful for assigning tasks to simulation objects without the need to edit a plan. You can assign them interactively, so simulation participants can respond immediately to events in a simulation by assigning new tasks to a simulation object as they might have to do in the field. General issues for assigning tasks are described in detail in "Chapter 29. Assigning Tasks" on page 557. Specific task procedures are in the succeeding chapters.

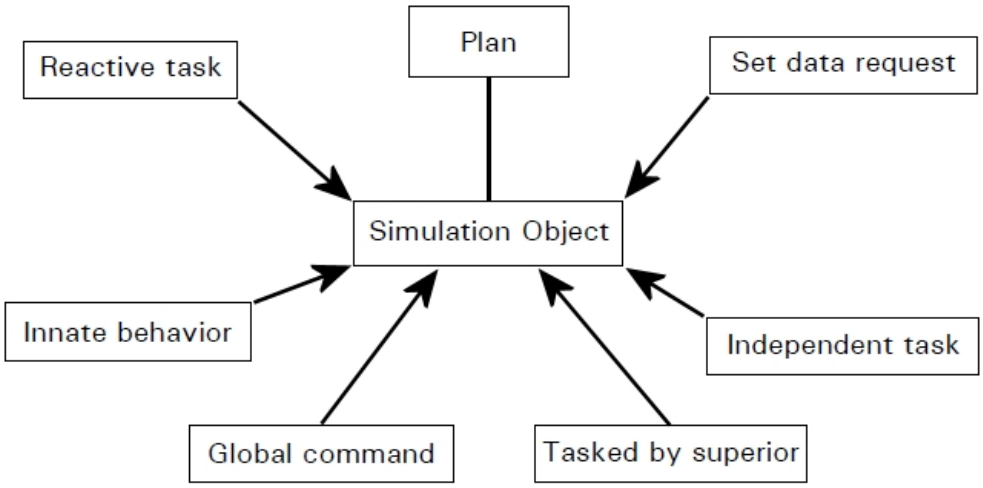
Many tasks are implemented as part of the VR-Forces code (C++ tasks) and cannot be added to or changed by end-users. Other tasks, called scripted tasks, are implemented using Lua scripts. You can edit scripted tasks and write new ones. For details, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743.

VR-Forces has a special type of scripted task, called a reactive task, that is not explicitly assigned to simulation objects. Reactive tasks monitor the simulation and run only if certain conditions are met. For information about reactive tasks, see "29.5. Reactive Tasks" on page 572.

A third type of scripted task is the background process. Background processes run continuously to handle certain types of ongoing aggregate-level activities, such as consuming or receiving supplies, calculating a unit's footprint, and automatically attacking opposing forces. For information about background processes, see "37.11. Background Processes" on page 812.

Figure 217 illustrates the many ways that a simulation object can have its task, behavior, and state changed.

Figure 217. Inputs to simulation object actions





29. Assigning Tasks

This chapter provides general details about how to assign tasks to simulation objects. For the procedures for assigning tasks, please see the rest of the chapters in this section.

29.1. Assigning Tasks to Simulation Objects 558
29.1.1 The Command Panel 558
29.1.2 Task Procedures 560
29.1.3 C++ Tasks and Scripted Tasks 560
29.1.4 Concurrent Task Execution 561
29.1.5 How Do I Know Which Simulation Objects Can Execute a Task? 562
29.1.6 Escaping the Task Assignment Process 563
29.1.7 Specifying Parameters for Tasks 563
29.1.8 Viewing Task Status 565
29.1.9 Filtering the Object Selection Lists 565
29.1.10 Skipping (Stopping) a Task 566
29.2. The Entity Control Panel 567
29.2.1 Specifying Entity Movement Using the Entity Control Panel 568
29.2.2 Using the Buttons in the Entity Control Panel 569
29.3. Using a Keyboard Shortcut to Assign a One-Click Task (or Set) 571
29.4. Assigning Tasks to Units 572
29.4.1 Convoy Tasks 572
29.4.2 Independently Tasking Unit Members 572
29.5. Reactive Tasks 572
29.5.1 Enabling Reactive Tasks 574
29.5.2 Disabling Reactive Tasks 574
29.5.3 Setting the Priority of a Reactive Task 575
29.5.4 Managing Reactive Tasks 576
29.5.5 Canceling a Reactive Task 576
29.6. Using Behavior Sets to Manage Scripts 577
29.6.1 Creating Behavior Sets 578

29.6.2 Editing Behavior Sets 579
29.6.3 Assigning a Behavior Set to a Force 580
29.7. Configuring Task Execution Rules 580
29.7.1 Configuring Action Categories 581

29.1. Assigning Tasks to Simulation Objects

You can assign tasks to a simulation object as part of its plan, as part of a global plan, and independently of any plan. For conceptual information about tasks, see "Introduction to Tasks" on page 554. For information about how to add a task to a plan, see "43.1.1 Adding a Task or Set Data Request to a Plan" on page 949.

You cannot assign tasks to non-VR-Forces simulation objects. However, you can use a remote simulation object as a parameter in a task, for example, follow or target a remote simulation object. For more information, see "42.5.6 Using Non-VR-Forces Simulation Objects in Plans" on page 945.

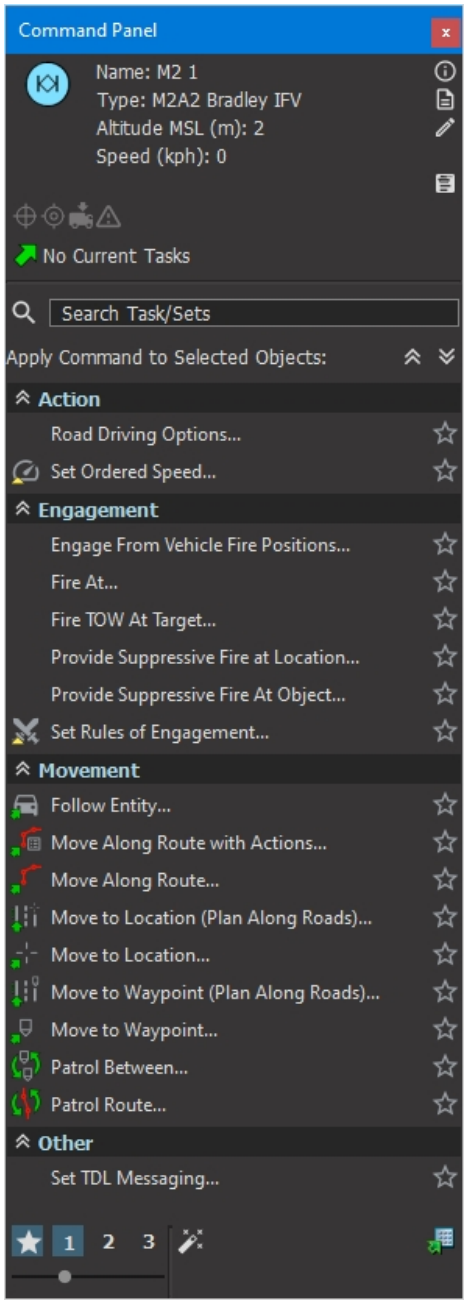
29.1.1 The Command Panel

The Command Panel brings together all of the tasks and sets available to a simulation object, along with basic status information, and presents them in a categorized menu that you can control to display more or fewer options. You can use it both to assign independent tasks during run time and to build plans.

By default, the Command Panel (Figure 218) opens on the right side of the VR-Forces window.

- ➤ To display the Command Panel if it does not open automatically—or to close it—choose **View > Command Panel**.
- ➤ To populate the Command Panel, select a simulation object in any observer mode or in the Objects List.

Figure 218. The Command Panel populated with commands for an M2A2 Bradley IFV



➤ Use the controls on the Command Panel to display more or fewer task and set options:

- Click the star (☆) next to a task or set to make it a favorite that is always visible.
- Click the star, 1, 2, or 3 at the bottom of the panel to display favorites (★), most common (1), advanced (2), or detailed (3) control commands.
- Click the magic wand (🪄) at the bottom of the panel to display options that are not possible in the real world (such as Set Location, which can move an entity to a distant location instantly).

- To create a plan for the selected simulation object or open the existing plan, click the plan icon () near the top of the Command Panel. For instructions on creating and editing plans, see "Chapter 43. Writing Plans" on page 947.
- Click any command on the panel to view the parameters for that task or set data request.

See the subsections that follow for general information on assigning tasks. For details on particular tasks, see the remaining parts of this Tasks section: "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583, "Chapter 31. Engagement Tasks for Entity-Level Scenarios" on page 623, "Chapter 32. Human and Crowd Tasks" on page 649, and so on.



To see icons instead of text for the tasks and set data requests, choose **Settings > Application**, select the GUI Settings page, and then select the Show Command Panel Icons check box. You can then choose whether to view them as small or large icons.

29.1.2 Task Procedures

For the procedures for assigning tasks, see "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583, "Chapter 31. Engagement Tasks for Entity-Level Scenarios" on page 623, "Chapter 32. Human and Crowd Tasks" on page 649, and "Chapter 34. Embarkation, Wait, Radio, and Other Tasks" on page 679.

If your installation uses the VR-Forces API to add additional tasks, consult with your software engineers for procedural information. See "24.1. Fixed-Wing Entity Behaviors" on page 510 and "24.2. Rotary-Wing Entity Behaviors" on page 513 for additional information about assigning tasks to these simulation object types.

The instructions for creating tasks direct you to use options on the Task menu. However, all the options available on the main menu are also available on context-sensitive menus and the Command Panel. A limited number of tasks are available on the Tasks Toolbar and on the Last Selected Object Panel.

29.1.3 C++ Tasks and Scripted Tasks

The VR-Forces application includes a basic set of tasks for movement, weapons fire, and other actions, that were developed using the VR-Forces Toolkit and the C++ API. System integrators often use the C++ API to add additional tasks before delivering VR-Forces to their customers. These tasks are called C++ tasks to distinguish them from another type of task, the scripted task. Scripted tasks, written in the Lua programming language, use the lower-level C++ tasks and the VR-Forces Lua API to build higher-order tasks for simulation objects. You do not need the VR-Forces Toolkit or a developers license to write scripted tasks.

To the VR-Forces end user, scripted tasks, most of which are listed on the Task menu, are indistinguishable from C++ tasks. For information about scripted tasks, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743.

VR-Forces also supports two special categories of scripted tasks called reactive tasks and background processes. These tasks are not on the Task menu. Reactive tasks execute

when a condition becomes true. For details, see "29.5. Reactive Tasks" on page 572. Background processes run continuously. (For details, see "35.9. Background Process List" on page 740.)

29.1.4 Concurrent Task Execution

Some tasks can execute at the same time. Some are mutually exclusive. If you assign a simulation object a task that is mutually exclusive with its current task, the current task is abandoned. If you assign a task that is not mutually exclusive with the current task, it executes the new task while continuing to execute the previous task.

When you assign a mutually exclusive independent task, VR-Forces displays a prompt that asks you to confirm interruption of the current task. You can disable this prompt if you always want a new task to interrupt the current task. For details, see "Enabling and Disabling the Task Confirmation Prompt" below.

C++ tasks are organized into these groups:

- Weapon
- Movement
- Radio
- Depth control

All tasks within a group are mutually exclusive among themselves. For example, all movement tasks are mutually exclusive. Simulation objects that are moving can usually execute a task from one of the other groups at the same time. For example, simulation objects can execute the various Send tasks while they are executing movement tasks. When you create a scripted task, you can specify which groups it conflicts with. For information about how to configure mutually exclusive tasks, see "29.7. Configuring Task Execution Rules" on page 580. For information about scripted tasks, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743. For information about how to use concurrent tasks in plans, see "42.5.2 Using Concurrent Tasks in Plans" on page 945.

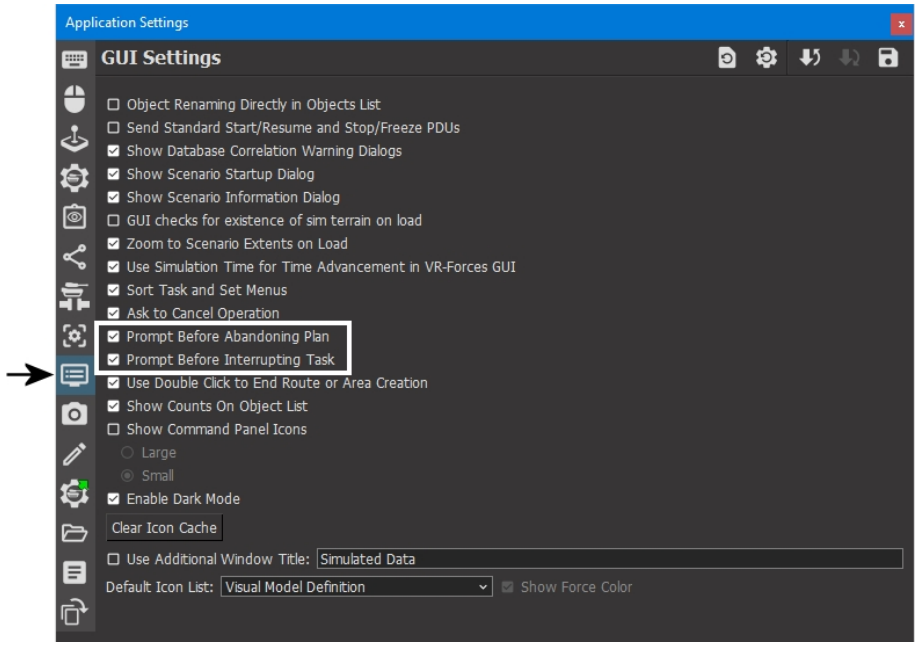
Enabling and Disabling the Task Confirmation Prompt

When you give a simulation object an independent task, if it is executing a task or a plan, VR-Forces informs you that the new task will override the task or plan and asks you to confirm the action. You can enable and disable the confirmation prompt.

➤ **To enable or disable the task override confirmation prompt:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 219).

Figure 219. GUI Settings page



3. To enable or disable the prompt for abandoning a plan, select or clear the Prompt Before Abandoning Plan check box.
4. To enable or disable the prompt for interrupting a task, select or clear the Prompt Before Interrupting Task check box.
5. Click Close.

29.1.5 How Do I Know Which Simulation Objects Can Execute a Task?

Some VR-Forces tasks apply to almost all simulation objects, such as Radio tasks and basic movement tasks. Others are valid only for particular platforms or even particular entity types. Sometimes it is obvious which entity types can execute a task. For example, you would expect that Rotary Wing Land only applies to rotary-wing entities. Similarly, Raise Periscope applies to submarines. However, for other tasks, it is not always evident which simulation object can execute a task.

Simulation objects can execute tasks for which they have an appropriate controller. They access most task controllers through systems. Therefore, simulation objects can execute the tasks supported by their various movement, weapon, sensor, and other systems. For example, to deploy naval mines, an entity needs a Naval Mine Deployment system.

To see which systems a simulation object has, and thereby infer which tasks it can execute, you can examine the simulation object's systems in the Simulation Object Editor. You can also refer to the *VR-Forces Entity Model Catalog*, which lists the systems for each simulation object configured in the Simulation Object Editor. "Appendix E. Systems and System Usage" on page 1693 lists each system and the scripted tasks associated with it.

29.1.6 Escaping the Task Assignment Process

You can escape the task assignment process at any point before you complete it.

➤ To escape the task assignment process, in the task parameters, click Cancel.

29.1.7 Specifying Parameters for Tasks

In VR-Forces, you specify task parameters in fields, dialog boxes, or by selecting tactical graphic or a simulation object on the terrain. When you select the appropriate object, the selection is reflected in the task parameters.

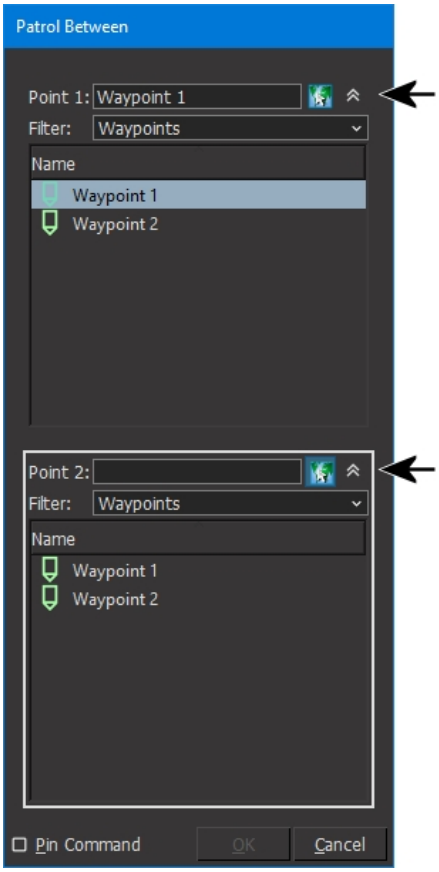
If you prefer, you can specify all parameters in the task parameters. Depending on the zoom level of the terrain and the number of simulation objects and tactical graphics you have created, it may be easier to select an object from a list than to select it from a densely packed group of icons.

Selecting Objects in Multiple List Parameters

Some tasks' parameters, such as those for Patrol Between, require you to select multiple objects, each of which is selected from a different list.

You can select the objects in the list window or in the display window. At any given moment, one of the lists is the active list and a selection in the display window is reflected in that list. The active list is indicated by a white rectangle around the list and the button next to the selection box (Figure 220).

Figure 220. Multiple list dialog box



The active list is determined as follows:

- When you initially open the parameters, the first list window is automatically active. If you select an object in the display, it is selected in the first list window. Then the second list window automatically becomes active and the next object you select in the display is selected in the second list window. This sequence continues until all list windows have a selection. Then the last window stays as the active list.
- Selecting an object in a list window makes it active.
- Clicking the Select Object button ( ) beside the box makes that list window active. It stays active until you click the button to deactivate it or click in another list window.
- If all lists are inactive (by clicking the Select Object button to put it into the unselected state), you can click objects in the display without affecting the selections in the list windows.

Pinning Task Parameters

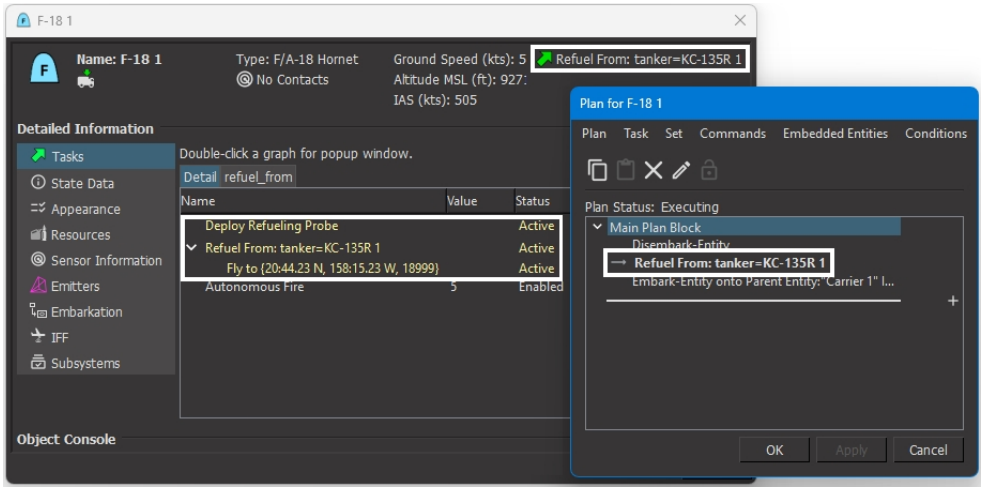
Use the Pin Command check box to pin the task parameters in the dialog box or the Command Panel. The OK button changes to an Execute button, and the entity will perform the task but the parameters will remain open. This allows you to quickly reassign the same task.

This option is also available for set data requests.

29.1.8 Viewing Task Status

The Information dialog box for a simulation object lists its current task on the summary section. It also lists the state of the reactive tasks that are enabled for the simulation object. If a simulation object is executing a task as part of its plan, you can see which task it is executing by opening its Plan window (Figure 221). You can also see the current task on the Last Selected Object Panel and the Tasks page of the Information dialog box.

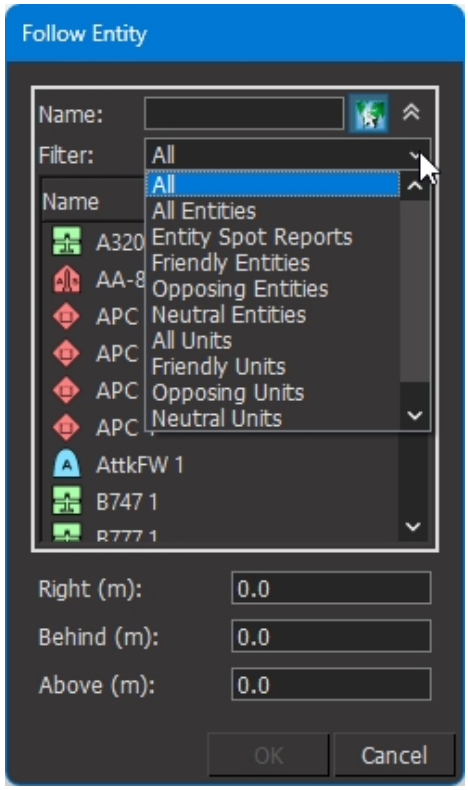
Figure 221. Display of simulation object task status



29.1.9 Filtering the Object Selection Lists

If a task requires you to select objects, you can filter the object selection list to make it easier to find the particular objects you need to select (Figure 222). If you are selecting simulation objects, for example, in a Follow Entity task, you can display all simulation objects, individual simulation objects only, or units only. Within the latter two groups, you can display friendly or opposing simulation objects. In a Patrol Between task, you can filter for waypoints, point tactical graphics, and plan variables.

Figure 222. Simulation object filter list



➤ To filter a simulation object list, in a task or set data request parameters, choose the filter type from the list.

29.1.10 Skipping (Stopping) a Task

You can instruct a simulation object to stop executing its current task. If the simulation object is executing a plan, it moves to the next task in its plan. If there are no further tasks, the simulation object waits for further orders.

➤ To order a simulation object to skip a task, choose **Objects > Manage Plan > Skip Task**.

- Skip Task does not apply to members of a unit that are executing tasks assigned to them by the unit leader.
- You can also stop tasks from the Task Status page of the Information dialog box. For details, see "29.5.5 Canceling a Reactive Task" on page 576.

29.2.  The Entity Control Panel

The Entity Control Panel is specific to a single entity. You can create one for each entity that you want to control.

For aircraft, surface watercraft, or subsurface watercraft, you use the panel to quickly assign:

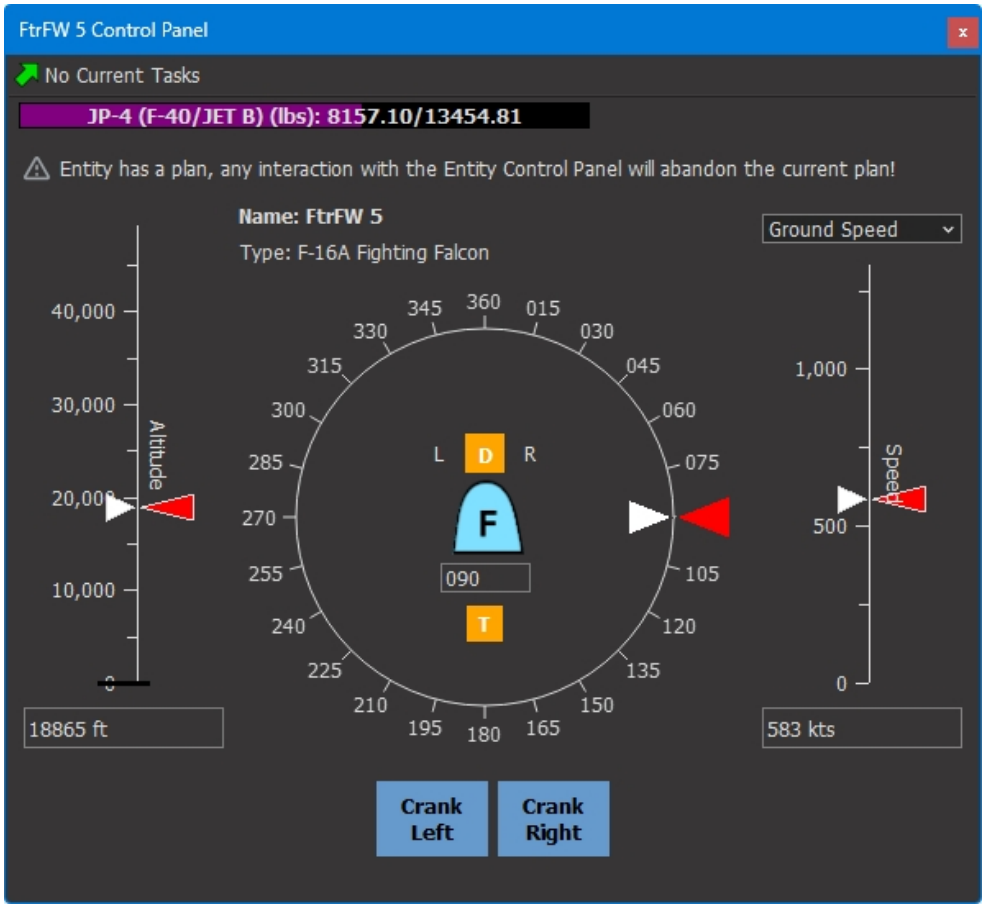
- Fly Heading and Altitude task (aircraft only)
- Sail Heading task (surface and subsurface watercraft only)
- Move to Depth task (subsurface watercraft only)
- Ordered Speed set data request

Using the configurable buttons at the bottom of the panel, you can assign entities other tasks or set data requests.

The Entity Control Panel displays the same information for ground entities and lifeforms as it does for aircraft and marine entities, but you can control them only by using the configurable buttons.

The Entity Control Panel (Figure 223) displays the name, type, altitude, heading, and speed of the selected entity with white arrows. Text boxes also show the current values. Red arrows show the desired values for aircraft, surface, and subsurface entities. The panel also shows the entity's current task.

Figure 223. Entity Control Panel



Using the fly or sail heading, fly altitude, move to depth, and ordered speed controls that comprise most of the Entity Control Panel do not interfere with any plans that the entity might have. The buttons at the button can cause the entity to abandon its current plan. You are given a chance to confirm whether that is what you want.

29.2.1 Specifying Entity Movement Using the Entity Control Panel

After opening an Entity Control Panel for an aircraft, surface watercraft, or subsurface watercraft, you can use the arrows and text boxes to specify its movement.

When you change the altitude or heading of those types of entities using the panel, you are issuing a movement task. Like any other task assignment, this action causes the entity to stop its current task and begin executing the new task.

By default, when you change heading, VR-Forces chooses the direction in which to turn. However, you can specify which direction you want it to turn.

You can specify that the heading be aligned with either true north or magnetic north. For information about magnetic north in VR-Forces, see "60.9.1 Specifying True, Magnetic, or Grid North" on page 1199.

If you change the ordered speed, the entity accelerates or decelerates to the desired speed. Changing the speed does not change the task.

When you single-click to change an entity's heading, speed, or altitude, then the entity makes the change at a normal speed. If you double-click, this indicates an emergency maneuver and the entity makes the change more rapidly than normal.

➤ **To assign a movement task from the Entity Control Panel:**

1. In the object list or terrain, select one or more entities to command.
2. Choose **Objects > Entity Control Panel** The Entity Control Panel opens for each selected entity.

   > You can also open the control panel for an entity by clicking Show Entity Control Panel () on the Last Selected Object panel.

3. To change altitude or depth, click along the Altitude line, click and drag the red arrow, or type an altitude in the text box.
4. To change heading, click a heading on the compass, click and drag the red arrow, or type a heading in the text box. To have the entity change heading at the maximum speed possible, double-click the heading. To specify the direction of turn, click L (left) or R (right).
5. The button below the heading text box indicates whether the heading is true north (T) or magnetic north (M). Click the button to change the setting.
6. To set the ordered speed, click along the Speed line, click and drag the red arrow, or type a value in the text box. For aircraft, you can use the list at the top right to specify the measurement of the speed: Ground Speed, TAS (true airspeed), IAS (indicated air speed), or Mach. You cannot set a speed greater than the entity's maximum speed.

29.2.2 Using the Buttons in the Entity Control Panel

You can click a button at the bottom of the Entity Control Panel to quickly assign tasks and set data requests to an individual entity. These buttons are configurable.

Assigning a Task or Set

➤ **To assign a task or set from the Entity Control Panel:**

1. In the object list or terrain, select one or more entities to command.
2. Choose **Objects > Entity Control Panel** The Entity Control Panel opens for each selected entity.

   > You can also open the control panel for an entity by clicking Show Entity Control Panel () on the Last Selected Object panel.

3. Click the button for the task or set that you want to assign.

Configuring the Buttons

One file determines what buttons are available: *./data/simulationModelSets/EntityLevel/gui/unifiedControlToolbar.xml*. VR-Forces includes some buttons by default. Only the buttons that are relevant to the entity display in its Entity Control Panel. For example, the Move To task displays for all entities but Patrol Area is only available to watercraft.

The default configuration for the buttons in the Entity Control Panel is:

```
<DtUnifiedControlToolbar buttonwidth="70" buttonheight="48"
spacing="4" fontSize="10" fontBold="true" justify="Center">
  <item task="crank_left" icon="false" background="#6699cc"/>
  <item task="move-to" icon="false" background="#BAB86C"
title="Move"/>
  <item task="crank_right" icon="false" background="#6699cc"/>
</DtUnifiedControlToolbar>
```

Table 30 describes the available parameters for customizing a button on the Entity Control Panel.

Table 30. unifiedControlToolbar.xml parameters

Keyword	Description
`task` `set`	ID of the C++ or Lua task or set. (string, required)
`icon`	Whether the button should show the icon for the task or set (if available) instead of the text. The default is not to show the icon. (Boolean)
`background`	Hexadecimal color to use as the background for the button.
`title`	Text for the button that is different from the task or set's name (when not displaying the icon). (string)
`justify`	Justification for the text. You can specify `Left`, `Center`, or `Right`. Does not apply to icons; those are always centered.

➤ **To configure the buttons in the Entity Control Panel:**

1. Open *./data/simulationModelSets/EntityLevel/gui/unifiedControlToolbar.xml* in your preferred tool for editing XML files, such as Notepad++.
2. Enter a new item entry or edit an existing one as desired, using the format:

```
<item = [task="task_ID"|set="set_ID"] (icon="[true|false]")
   (background="#nnnnnn") (title="title")
   (justify="[Left|Right|Center]")/>
```

- There is no limit to the number of buttons that you can add, but do note that they will not wrap. The Entity Control Panel simply grows wider to accommodate the buttons.
- You cannot add multiple buttons for the same task or set.
- Any changes you make go into effect the next time you start VR-Forces. If you make changes while the application is running, you must shut it down and restart it to see the new button configuration.

29.3.  Using a Keyboard Shortcut to Assign a One-Click Task (or Set)

You can quickly task an individual entity with a "one-click" task. VR-Forces is configured with two such tasks: move to a location or fire on a target or location. These one-click tasks do not require you to enter any values: you simply click the entity on the terrain to select it (not in the Objects List), press the appropriate keyboard combination, and then click a simulation object or a location on the terrain.

The setting to allow users to assign a Lua script task (or set data request) with a keyboard shortcut and a mouse click is One Click Support. See "36.2. Creating a New Script" on page 747 to learn more about the parameters for Lua scripts.

➤ **To assign a one-click task:**

1. Select the entity.
2. Press the appropriate keyboard combination, for example:
   - **Ctrl+M** to move to a location.
   - **Ctrl+F** to fire on a target or location.
3. Click the location or simulation object on the terrain or in the Objects List. While the scenario is running, the entity immediately performs the task (or set) if able.

- When using the keyboard shortcut, the default is for information to display on the screen, for example, `Move To: Destination`. If you do not want to see the text, clear the Map Status Text or Task/Set Status check boxes in the Observer Settings page of Display Settings (Figure 448 on page 1165).
- Note that these shortcuts only allow you to task entities to move to or fire at anything that they are configured for. It does not override any restrictions. For example, an entity configured only to fire at simulation objects cannot fire on a location.
- You can use shortcuts for units only when the unit object is not hidden (collapsed). See "19.2. Expanding and Collapsing Units" on page 444.

29.4. Assigning Tasks to Units

➤ To assign a task to a unit, select the unit, then assign it a task as you would any individual entity.

29.4.1  Convoy Tasks

VR-Forces has two tasks for moving in convoys: Convoy Along and Convoy To. These tasks can only be assigned to units made up of ground vehicles. When a unit receives a convoy task, it organizes itself into a column based on the echelon IDs. Then it executes the task.

Once the task begins, the members of the unit try to maintain a consistent distance apart from each other. You can configure this distance by editing the Ground Convoy Movement system. The distance is specified by the **separation-distance** parameter. You can also specify an amount by which the separation distance can vary using the **separation-tolerance** parameter.

> If VR-Forces determines that a convoy task is not appropriate for a given unit—for example, a rotary-wing unit—the convoy tasks are not available on the Task menu or in the Command Panel. In other cases, however, such as mixed units or when you aggregate disparate entities as Ground Unit, it is up to you to know whether a unit consists only of ground vehicles or contains platforms that do not support convoy tasks.

29.4.2 Independently Tasking Unit Members

Members of a unit can have individual plans or be assigned tasks independently of the unit. If a member of a unit is independently tasked, once it completes its task, it then automatically becomes tasked by its superior. You do not need to give it a Tasked by Superior set data request. You can override this behavior by setting an object's **tasked-by-superior-upon-task-complete** parameter in the object parameter database to False.

If a Skip Task command is given to an independently tasked subordinate, it remains in the independently tasked state until it either completes a task or it is given a Set Tasked by Superior set data request.

When a simulation object becomes tasked by its superior, it does not execute the task that the unit is currently executing. It receives the next task that is sent out by the unit.

29.5. Reactive Tasks

Reactive tasks are a special category of scripted tasks. Unlike other tasks (whether scripted or C++), reactive tasks do not begin to execute as soon as they are assigned. They monitor events in a simulation and execute in response to conditions specified in the script. In this way they are similar to collision avoidance, which takes over entity movement in response to an imminent collision, or trigger statements in plans, which are triggered when a condition becomes true.

Because reactive tasks are scripted tasks, you are not restricted to the few conditions available in plans. You have the full VR-Forces Lua API available to script the conditions and the resulting behaviors. Like other scripted tasks, reactive tasks are either scenario-specific or system reactive tasks.

Reactive tasks have the following behaviors and characteristics:

- Reactive tasks can be automatically enabled for a category of simulation object. They can also be individually enabled and disabled.
- When a reactive task activates, the current task is suspended. When the reactive task concludes, the simulation object resumes the previous task, if possible. There may be cases where the previous task is no longer meaningful due to changes in the scenario.
- Reactive tasks support task concurrency.
- Reactive tasks can be interrupted by other reactive tasks. Precedence among reactive tasks is based on a task's priority. If a reactive task is executing and a reactive task with a higher priority triggers, it suspends the current reactive task. When the higher priority task completes, the lower priority task resumes. When that task completes, the original task resumes. Multiple nesting of tasks in this way is permitted. Reactive tasks with the same or lower priority do not interrupt the active task.
- Reactive tasks can have parameters, just like any other task. If a reactive task is enabled and the parameters are not set, it may not work as expected. A well-written reactive task should have default values for parameters, but VR-Forces does not enforce this.
- Users can change the priority of a reactive task while a scenario is running. This affects the next activation of the task.
- If a reactive task is interrupted by a non-concurrent independent task assignment from the Task menu, the Command Panel, or a global plan, the reactive task and all suspended tasks are canceled and the new task is executed. However, it is still possible that the new task could be interrupted by the same or a different reactive task.
- Active reactive tasks can be canceled by a VR-Forces user.
- After a reactive task is complete, it enters the enabled state and can become triggered again.
- The status of a reactive task displays on the Tasks page of the Information dialog box, along with the status of the current or suspended task.

The following sections describe how to manage reactive tasks in a scenario. VR-Forces includes a collection of reactive tasks, listed in "34.14. Reactive Tasks" on page 707. For information about how to write and configure reactive tasks, see "Chapter 37. Writing Scripts" on page 783.

29.5.1 Enabling Reactive Tasks

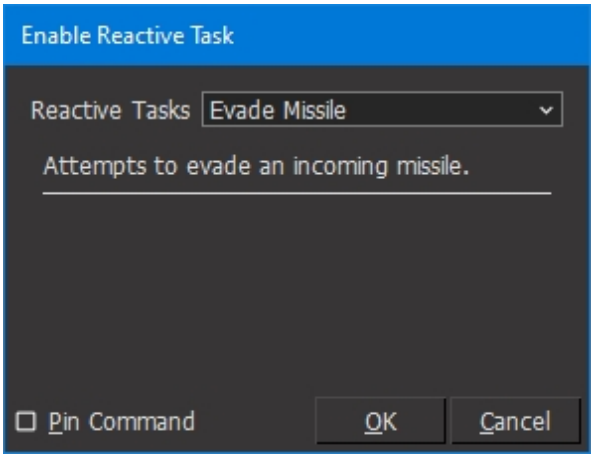
Like other scripted tasks, when a reactive task is written, it is specified as being valid for one or more simulation object types. It can also be configured to be enabled by default. If a reactive task is not enabled by default, you can enable it for specific simulation objects.

> This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

➤ **To enable a reactive task:**

1. Select the simulation object for which you want to enable the task.
2. Choose **Set > Reactive Tasks > Enable**. The Enable Reactive Task dialog box opens (Figure 224). If there are no reactive tasks available, the dialog box states this. This condition occurs if all appropriate reactive tasks are already enabled or there are no reactive tasks for this entity type.

Figure 224. Enable Reactive Task dialog box



3. In the Reactive Tasks list, select the task you want to enable. If the task requires parameters, the dialog box redisplays and lists the parameters.
4. If necessary, specify the parameters.
5. Click OK.

> You can also enable reactive tasks in the Manage Reactive Tasks dialog box. For details, see "29.5.4 Managing Reactive Tasks" on page 576.

29.5.2 Disabling Reactive Tasks

You can disable reactive tasks on a per-simulation object basis. When you disable a reactive task, any active tasks continue to execute. However, no new instances of the disabled task will be activated.

> This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

➤ **To disable a reactive task for a simulation object:**

1. Select the simulation object for which you want to disable a reactive task.
2. Choose **Set > Reactive Tasks > Disable**. The Disable Reactive Task dialog box opens. If there are no reactive tasks available, the dialog box states this. This condition occurs if all appropriate reactive tasks are already disabled or there are no reactive tasks for this simulation object type.
3. In the Reactive Tasks list, select the task you want to disable.
4. Click OK.

> You can also disable reactive tasks in the Manage Reactive Tasks dialog box. For details, see "29.5.4 Managing Reactive Tasks" on the next page.

29.5.3 Setting the Priority of a Reactive Task

When a reactive task is created, it is assigned a priority. The priority determines which reactive tasks it can override or be overridden by. Priorities are expressed as integers, with 1 being the highest priority. You can change the priority of a reactive task on a per-simulation object basis.

> This procedure applies to the currently selected simulation object. If you have multiple Information windows open, selecting a reactive task on the Task Status page of a simulation object does not select the simulation object and does not make it the task affected by this procedure.

➤ **To change the priority of a reactive task:**

1. Select the simulation object for which you want to change a reactive task's priority.
2. Choose **Set > Reactive Tasks > Priority**. The Reactive Task Priority dialog box opens. If there are no reactive tasks available, the dialog box states this.
3. In the Reactive Tasks list, select the task whose priority you want to change.
4. Select a priority in the box. The lower the number, the higher its priority.
5. Click OK.

> You can also change priority in the Manage Reactive Tasks dialog box. For details, see "29.5.4 Managing Reactive Tasks" on the next page.

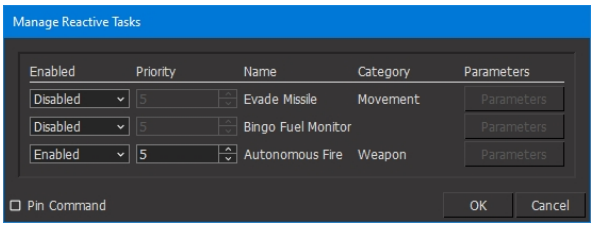
29.5.4 Managing Reactive Tasks

You can enable, disable, and change priority for all the reactive tasks that apply to a simulation object from the Manage Reactive Task dialog box.

➤ **To manage multiple reactive tasks for a simulation object:**

1. Select the simulation object for which you want to manage reactive tasks.
2. Choose **Set > Reactive Tasks > Manage** or choose **Objects > Manage Reactive Tasks**. The Manage Reactive Task dialog box opens (Figure 225). It lists each reactive task configured for this simulation object type, its status, and its priority. If a task requires parameters, you can set them.

Figure 225. Manage Reactive Task dialog box



3. For each task, to enable or disable, select an option from the list in the Enabled column.
4. For each task, change the priority by selecting an option in the box in the Priority column.
5. For each task that has parameters, click Parameters and set its parameters. (A task must be enabled to set its parameters.)
6. Click OK.

29.5.5 Canceling a Reactive Task

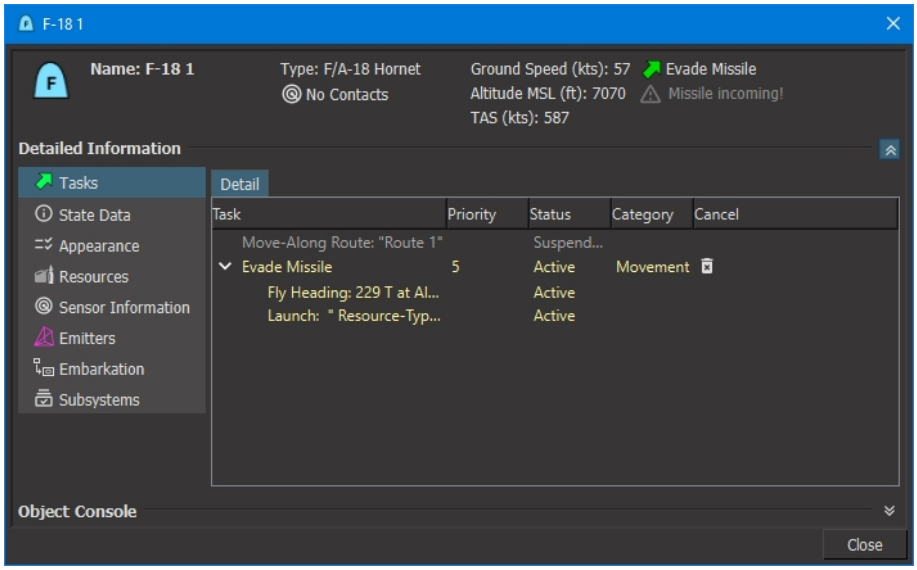
You can cancel a reactive task while it is executing. When you cancel a reactive task, the simulation object resumes the suspended task. However, if the condition that caused the reactive task to activate is still true, when you cancel a reactive task, it will usually immediately activate again.

ⓘ You can also use this procedure to cancel (skip) a regular task.

➤ **To cancel a reactive task:**

1. Select the simulation object whose task you want to cancel.
2. Choose **Objects > Information**, or press **I**. The Information dialog box opens.
3. Select the Tasks page. If a reactive task is active, it has a button in the Cancel column (Figure 226).

Figure 226. Active reactive task



4. Right-click the active task and choose **Cancel Reactive Task**, or click the button in the Cancel column.

29.6. Using Behavior Sets to Manage Scripts

Behavior sets let you organize scripts so that a set of related tasks and set data requests can be enabled or disabled at the force level.

If a script is not assigned to a behavior set, its availability to a simulation object is determined by the following settings:

- The Valid for Entity Types list for the script.
- For tasks specific to systems, configuration of that system on a simulation object.
- For reactive tasks, the Enable/Disable setting in the Manage Reactive Tasks dialog box for the simulation object.

If a script is assigned to a behavior set, then in addition to the requirements in the previous list, it is available to simulation objects only if the behavior set is assigned to a force and only to simulation objects of that force. (You can assign a behavior set to more than one force.)

As an example of how you might use behavior sets, suppose that dismounted infantry react differently to an attack depending on whether they are operating in rural terrain or in urban terrain. You have a set of scripted tasks that you have written for reacting to attack in each of these two situations. If you are not using behavior sets, then to manage which tasks are used when the simulation objects are under attack, you might use one of these strategies:

- Tell users not to use the inappropriate tasks.
- Enable and disable the correct set of tasks in the Scripts dialog box before the start of the scenario.
- Enable and disable the tasks on a per simulation object basis in the Manage Reactive Tasks dialog box.

Shifting back and forth between simulations in different terrains might become a management problem.

Using behavior sets, you could:

1. Create a behavior set for rural terrain operations and one for urban operations.
2. Assign the relevant scripts to each behavior set.
3. Before you run a scenario, you would assign the behavior set you want to use to the force.

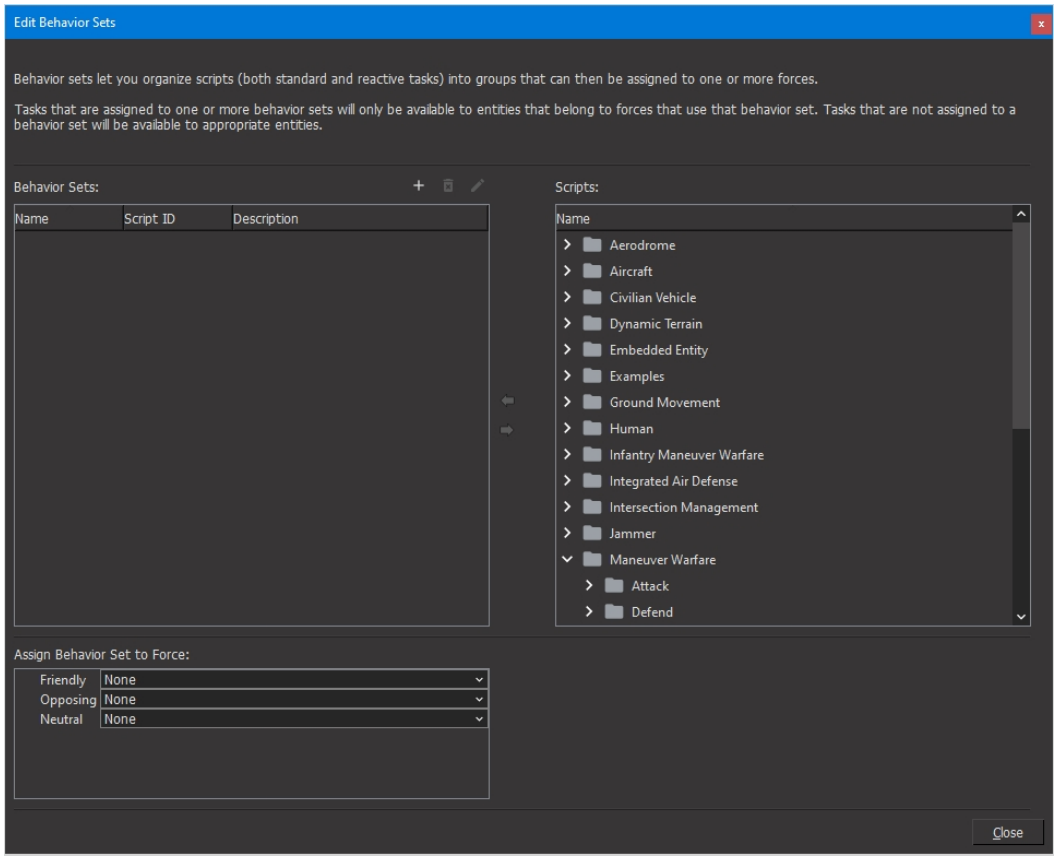
The tasks in the behavior set would be enabled and those in the unused behavior set would be disabled. There would be no further configuration required.

29.6.1 Creating Behavior Sets

➤ **To create a behavior set:**

1. Choose **Simulation > Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 227).

Figure 227. Edit Behavior Sets dialog box



2. Click the Add button (+). The New Behavior Set Name dialog box opens.
3. Enter a name for the behavior set.
4. Click OK.
5. In the Scripts list, select the scripts that you want to assign to this behavior set.
6. Click the left-facing arrow. The scripts are added to the behavior set. If you select a folder, all tasks in the folder are added to the behavior set. (You can also assign tasks to behavior sets in the Edit Script dialog box. For details, see "36.2. Creating a New Script" on page 747.)
7. Optionally, assign the behavior set to a force.

29.6.2 Editing Behavior Sets

You can edit behavior sets, rename them, and delete them.

➤ **To edit behavior sets:**

1. Choose **Simulation > Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 227).
2. Expand the behavior set that you want to edit.

3. Add and remove scripts by selecting them and clicking the appropriate arrow button.

➤ To rename a behavior set, select it in the Behavior Sets list, click the Rename button () and then enter a new name.

➤ To delete a behavior set, select it in the Behavior Sets list and click the Delete button ().

29.6.3 Assigning a Behavior Set to a Force

You can only assign one behavior set to a force at any given time.

➤ **To assign a behavior set to a force:**

1. Choose **Simulation > Behavior Sets**. The Edit Behavior Sets dialog box opens (Figure 227).
2. In the Assign Behavior Set to Force window, click the list for the force to which you want to assign a behavior set.
3. Select the behavior set that you want to assign. If you do not want any behavior sets assigned, select the blank line in the list.

29.7. Configuring Task Execution Rules

Each simulation object OPE file contains a parameter called **task-execution-rules** that specifies a task execution rule configuration file. (The default rules file is *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*.) The task execution rule file specifies whether tasks can be executed concurrently, and if so, which tasks are mutually exclusive.

The task execution rules file has these parameters:

- **all-tasks-exclusive**. This parameter enables (False) or disables (True) use of concurrent execution of tasks. When True, the Task Manager executes only one task at a time, and the **mutually-exclusive-task-groups** parameter is ignored.
- **mutually-exclusive-task-groups**. This parameter is a list of groups. Each group contains a list of tasks that cannot be executed at the same time. Any time a task from a particular group is started by the Task Manager, any currently running task that is part of the same group is interrupted. Tasks that are part of other groups, or not part of any group, are allowed to continue running.

The task names that are listed in the **mutually-exclusive-task-groups** parameter are the names that are registered with the task factory when a task is added to the system. Groups can contain any number of tasks, and there can be any number of groups.

Tasks are always considered to be mutually exclusive with themselves. For that reason, it is impossible to execute two tasks of the same type at the same time on a single simulation object.

> *default-task-rules.tsk* configures the tasks that come with VR-Forces and which were developed in C++. It does not include scripted tasks. Concurrency for scripted tasks is set when they are created.

29.7.1 Configuring Action Categories

In scripted tasks, action categories serve the same function as task rules for enforcing concurrency. Action categories are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/actionCategories.tsk*. An action category is specified with the entry:

```
<item>
  <first>Movement</first>
  <second>
    <first>Movement</first>
    <second>1</second>
  </second>
</item>
```

If the value in the <first> element matches a group in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*, scripted tasks that use that action category are unioned with the tasks specified in *default-task-rules.tsk* to form the complete set of mutually exclusive tasks.

> If you edit *actionCategories.tsk*, be sure to update the <count> element to reflect additions or deletions.



30. Movement Tasks for Entity-Level Scenarios

This chapter describes the movement tasks supplied with VR-Forces that apply to entity-level scenarios. Some of these tasks are also used in aggregate-level scenarios. "Chapter 32. Human and Crowd Tasks" on page 649 describes some movement tasks that are specific to human characters. "33.5. Movement Tasks for Unit Behaviors" on page 673 describes movement tasks for more advanced unit behaviors. "Chapter 16. Generating Traffic (Pattern of Life)" on page 415 discusses methods of generating air, sea, and ground traffic to provide realistic background activity in your scenarios.

Models with wheels and treads have the ability to rotate those at speeds relative to the entity's movement speed. Wheels can also articulate when the entity makes a turn to realistically represent the steering.

30.1. Airdrop Along Route 585
30.2. Animated Movement 585
30.3. Come to Stop 586
30.4. Convoy Along 586
30.5. Convoy To 587
30.6. Crank Left and Crank Right 587
30.7. Establish Refueling Station 587
30.8. Fly Altitude 589
30.9. Fly Cloverleaf Pattern 589
30.10. Fly Figure 8 Pattern 590
30.11. Fly Heading and Altitude 591
30.12. Fly Heading 592
30.13. Fly Loiter Pattern 592
30.14. Follow Entity 593
30.14.1 How Fixed-Wing Entities Follow Entities 594
30.15. Hover 594
30.16. Keep Station 595
30.17. Land At Airport 596
30.18. Land at Airport Near Location 596

30.19. Land On Route 596
30.20. Land On Runway 597
30.21. Land (Vertical) 598
30.22. Maneuver Along 598
30.23. Maneuver To 599
30.24. Move Along Route 599
30.25. Move Along Route with Actions 600
30.26. Move Into Formation 603
30.27. Move Next to Target 603
30.28. Move To 604
30.29. Move to Altitude 605
30.30. Move to Bullseye Offset 605
30.31. Move to Depth 606
30.32. Move To (Plan Path) 606
30.33. Orbit 607
30.34. Orbit Object 607
30.35. Paratrooper Air Drop 608
30.36. Patrol Area 609
30.37. Patrol Between 609
30.38. Patrol Route 609
30.39. Pattern Hold 610
30.40. Refuel From 610
30.41. Return To Base 611
30.42. Rotary Wing Airlift Cargo to Location 611
30.43. Rotary Wing Land 612
30.44. Rotary Wing Retrieve Cargo 612
30.45. Rotary Wing Unload Cargo 612
30.46. Sail Heading 613
30.47. Search and Rescue 613
30.47.1 Fly Creeping Line Pattern 613
30.47.2 Fly Expanding Square Pattern 615
30.47.3 Fly Parallel Pattern 616
30.47.4 Sector Search 618
30.48. Sweep Naval Mines 619
30.49. Takeoff On Route 619
30.50. Takeoff On Runway 620
30.51. Takeoff (Vertical) 621
30.52. Transition Into Formation 621
30.53. Turn to Heading 622
30.54. Zig-Zag to Heading 622

30.1. Airdrop Along Route

The Airdrop Along Route task causes an aircraft to move along a specific route or to a specific plan variable. Upon reaching the route or variable, the aircraft disembarks paratroopers or entities such as vehicles. It specifies only when the entities embarked on the aircraft will parachute, and it can specify the number of "drop slots" available to those objects.

Drop slots are not the same as embarkation slots. On an aircraft, drop slots indicate the number of intervals at which entities can exit the aircraft with the Airdrop from Aircraft task. On entities, the drop slot indicates the order in which they will perform that task. Drop slots do not support units.

The entities must have a task for deploying the parachute or they will not exit the aircraft. See "34.1.5 Airdrop from Aircraft" on page 684 for details.

> The movement task Parachute to Target does not coordinate with Airdrop Along Route.

➤ **To assign an Airdrop Along Route task:**

1. Select an aircraft that has entities embarked on it.
2. Choose **Task > Movement > Airdrop Along Route**. The Airdrop Along Route parameters display.
3. Specify the time to wait between each entity drop in seconds.
4. Use Drop Route to specify where the aircraft should begin the airdrop. It can be a route or a plan variable.
5. Optionally, you can select the Use Drop Slots check box and then specify the number of drop slots. One or more entities can exit the aircraft at each drop slot, as specified in the Airdrop from Aircraft task.
6. Click OK.

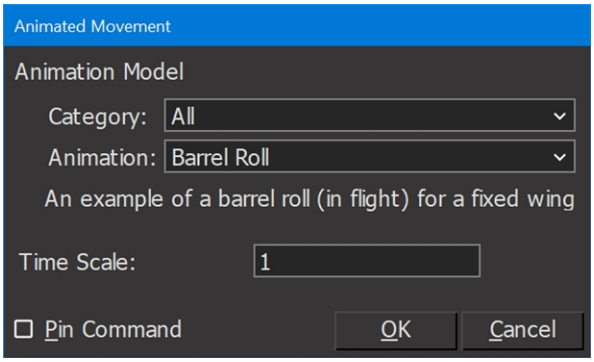
30.2. Animated Movement

Use the Animated Movement task to assign a scripted behavior to an entity. To use this task, you must first create a file that describes the movements that you want an entity to make. Animated movement files can be in comma-separated values (CSV) format or 3DS ASCII Export (ASE) format. You configure animated movement tasks in the Simulation Object Editor. For details, see "70.8.2 Configuring Animated Movement" on page 1401.

➤ **To assign an animated movement task:**

1. Select the entity to task.
2. Choose **Task > Movement > Animated Movement**. The Animated Movement parameters display (Figure 228).

Figure 228. Animated Movement dialog box



3. Optionally, filter the animation list by selecting a category in the Category list.
4. Select an animation in the Animation list. You are responsible for knowing that the animation is appropriate to the entity you selected.
5. Optionally, specify a time scale. By changing the time scale you can make the animation run faster or slower than specified in the animation file.
6. Click OK.

30.3. Come to Stop

The Come to Stop task stops a surface watercraft entity using a normal stop, an emergency stop, or a slow drift.

➤ **To assign a Come to Stop task:**

1. Select the entity.
2. Choose **Task > Movement > Come to Stop**. The Come to Stop parameters display.
3. Select an option for how to stop.
4. Click OK.

30.4. Convoy Along

The Convoy Along task commands a unit to move along a route as a convoy. When the task starts, the lead entity moves far enough along the route for the rest of the unit to create a column that ends approximately at the beginning of the route. Then the unit begins moving along the route.

ⓘ The Convoy Along task is valid only for ground-vehicle aggregates.

For more details about convoys, see "29.4.1 Convoy Tasks" on page 572.

➤ **To assign a Convoy Along task:**

1. Select the unit that you want to send in a convoy.
2. Choose **Task > Movement > Convoy Along**. The Convoy Along parameters display.

3. Specify the route to convoy along.
4. Click OK.

30.5. Convoy To

The Convoy To task commands a unit to move to a waypoint as a convoy. If you specify use of roads, VR-Forces uses the information in the road network to plan the convoy's movement. If you do not specify use of roads, the convoy moves directly across the terrain as an individual entity would move in a Move To task.

> The Convoy To task is valid only for ground-vehicle aggregates.

For more details about convoys, see "29.4.1 Convoy Tasks" on page 572.

➤ **To assign a Convoy To task:**

1. Select the unit that you want to send in a convoy.
2. Choose **Task > Movement > Convoy To**. The Convoy To parameters display.
3. Specify the waypoint to convoy to.
4. Optionally, select or clear the Use Roads check box.

   > When you specify use of roads, VR-Forces looks for roads over a fairly wide area and if one is found, will try to use it even if this results in an indirect approach to the waypoint. If there are no roads close to the route you would expect the unit to take to reach the waypoint, do not select this option.

5. Click OK.

30.6. Crank Left and Crank Right

The Crank Left and Crank Right movement tasks cause a fixed-wing aircraft to immediately turn to and fly a heading. The default is to change by 60 degrees, turning at 6 degrees per second.

➤ To assign the Crank Left or Crank Right task, select the entity and then choose **Task > Movement > Crank Left** or **Crank Right**. The entity changes its heading accordingly.

> You can also use the Crank Left and Crank Right buttons in the Entity Control Panel. See "29.2. The Entity Control Panel" on page 567.

30.7. Establish Refueling Station

Entities with a refueling system can establish a refueling station where other aircraft can refuel using the Refuel From task. The Establish Refueling Station task requires two waypoints: one to use as the Air Refueling Control Point (ARCP) and the other to use as the Air Refueling Initial Point (ARIP). These should be placed no farther than 50 kilometers apart in the terrain.

While performing the Establish Refueling Station task, the tanker aircraft first moves to the specified ARCP waypoint and then flies a holding pattern relative to that waypoint. The tanker then waits for aircraft to request refueling and queues the aircraft in the order it received the requests. It flies the holding pattern for a specified length of time.

An aircraft performing the Refuel From task (see "30.40. Refuel From" on page 610) then flies to the tanker's ARIP waypoint. When it is about four minutes away from the waypoint, that aircraft sends a request. The tanker either accepts or rejects the request. If it accepts the request, the refueling proceeds as described below. If it rejects the request, the tanker returns to its station and the aircraft performs the next task in its plan, if any. Reasons that the tanker would reject a request are:

- Its aviation-fuel resource is depleted.
- The aircraft has a different fuel supply type than what the tanker is carrying. The fuel supply type is identified by its SISO entity type.
- The aircraft requires a different refuel connection than is available on the tanker, either because the tanker is not equipped with it or it is disabled.

  Tankers can be equipped with an air-to-air refueling system: boom, boom wingtip drogue, wingtip drogue, centerline boom drogue attachment, or multi-point. Aircraft can be equipped with an air refueling receiver boom or drogue system.

If the tanker accepts the request, it sends a message to the aircraft that indicates a rendezvous location and an approximate time of arrival. The aircraft makes altitude and speed adjustments to arrive at the location on time. Once the two aircraft are near the ARCP waypoint, the tanker sends a refueling location to the aircraft. The aircraft then moves to that location and the tanker and aircraft fly in a straight line. The tanker deploys its air-to-air refueling system and the aircraft moves to connect with it. During the refueling, the aircraft's movement|fuel resource increases as the tanker's aviation-fuel resource decreases.

When the aircraft's fuel is nearly full, they separate. The aircraft moves to the next statement in its plan, if any. The tanker returns to refueling station holding pattern and checks for other refuel requests. If another aircraft placed a request while the tanker was refueling the first aircraft, that aircraft orbits the ARIP waypoint until its request is accepted or rejected.

You can also task a tanker to refuel itself. See "34.2. Internal Transfer of Fuel" on page 692.

➤ **To assign an Establish Refueling Station task:**

1. Select the entity.
2. Choose **Task > Movement > Establish Refueling Station**. The Establish Refueling Station parameters display.
3. Specify the ARCP waypoint.
4. Specify the ARIP waypoint.

5. Specify a heading, in degrees, and indicate whether it is relative to true or magnetic north.
6. Enter the speed at which the entity should travel.
7. Enter the altitude it should fly.
8. Select the desired Turn Direction.
9. Enter the length of time that the aircraft should spend at the refueling station waiting for refuel requests.
10. Specify the Pattern Type, either Time or Distance.
    - When using the Time pattern, you specify the length of time to complete each leg of the flight pattern.
    - When using the Distance pattern, you specify the length and width of the pattern.
11. Click OK.

30.8. Fly Altitude

The Fly Altitude task commands a fixed-wing or rotary-wing entity to fly to the specified altitude at a specified ascent or descent rate, using its current heading and ordered speed. When it reaches the altitude it continues to fly at that altitude at the current heading until it receives other commands. The entity will not exceed the limits for altitude or ascent/descent rate set in the OPD for the entity type.

You cannot use a Fly Altitude task to cause a fixed-wing entity to take off. A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it rises to the specified altitude as it moves forward at its current heading. If a rotary-wing entity is in the air and is given this task with an altitude of zero, it descends to approximately 33 M and continues flying at that altitude at its current heading.

➤ **To assign a Fly Altitude task:**

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task > Movement > Fly Altitude**, or click the Fly Altitude button (). The Fly Altitude parameters display.
3. In the Altitude box, specify the altitude above sea level to fly to.
4. In the Climb/Descent Rate box, specify the rate.
5. Click OK.

30.9. Fly Cloverleaf Pattern

The Fly Cloverleaf Pattern task commands an aircraft to fly in a cloverleaf. The entity flies around the point you specify as the center of the pattern. You specify the bearing and length to indicate the direction of the main axis and the space between loop centers. You also determine the turn direction, altitude, speed, and the duration of the pattern flight.

> If the entity begins the task close to the reference point that you specify as the start of the task, it may need to fly one extra loop to ensure that point is at the center of the pattern. This is determined by the radius of the loops and the length between the centers of the loops.

The entity will not exceed the limits on altitude, air speed, or maximum lateral acceleration set in the OPD.

➤ **To assign a Fly Cloverleaf Pattern task:**

1. Select a fixed-wing entity.
2. Choose **Task > Movement > Fly Cloverleaf Pattern**. The Fly Cloverleaf Pattern parameters display.
3. In the Loiter Point box, select the point tactical graphic that marks the center point for the cloverleaf pattern.
4. In the Radius box, specify the radius of the loops made in the pattern.
5. In the Bearing box, enter the bearing of the main axis of the pattern.
6. In the Length box, enter the space between loop centers.
7. Select the desired Turn Direction.
8. In the Altitude box, specify the altitude above sea level to fly the pattern.
9. In the Speed box, enter the speed to fly the pattern.
10. Specify the desired Duration Type:
    - Duration in Laps: Enter the number of laps the entity should fly the pattern. This must be a whole number.
    - Duration in Time: Enter the length of time that the entity should fly the pattern. The first box is the number of days, and the second box is hours, minutes, and seconds.
11. Click OK.

30.10. Fly Figure 8 Pattern

The Fly Figure 8 Pattern task commands an aircraft to fly in a figure eight. The entity flies around the point you specify as the center of the pattern. You specify the bearing and length to indicate the direction of the main axis and the space between loop centers. You also determine the turn direction, altitude, speed, and the duration of the pattern flight.

The entity will not exceed the limits on altitude, air speed, or maximum lateral acceleration set in the OPD.

➤ **To assign a Fly Figure 8 Pattern task:**

1. Select a fixed-wing entity.
2. Choose **Task > Movement > Fly Figure 8 Pattern**. The Fly Figure 8 Pattern parameters display.

3. In the Loiter Point box, select the point tactical graphic that marks the center point for the figure eight pattern.
4. In the Radius box, specify the radius of the loops made in the pattern.
5. In the Bearing box, enter the bearing of the main axis of the pattern.
6. In the Length box, enter the space between loop centers.
7. Select the desired Turn Direction.
8. In the Altitude box, specify the altitude above sea level to fly the pattern.
9. In the Speed box, enter the speed to fly the pattern.
10. Specify the desired Duration Type:
    - Duration in Laps: Enter the number of laps the entity should fly the pattern. This must be a whole number.
    - Duration in Time: Enter the length of time that the entity should fly the pattern. The first box is the number of days, and the second box is hours, minutes, and seconds.
11. Click OK.

30.11. Fly Heading and Altitude

The Fly Heading and Altitude task commands a fixed-wing or rotary-wing entity to fly to a specified altitude, at a specified ascent or descent rate, while turning to a specified heading at a specified turn rate. If you do not specify a heading or altitude, the entity continues to use the current value for that parameter.

While executing this task, the entity maintains its current speed. When it completes the command, it continues to fly at the new altitude and heading until given other commands. The entity will not exceed the limits on altitude, turn rate, and ascent/descent rate set in the OPD.

You cannot use a Fly Heading and Altitude task to cause a fixed-wing entity to take off. A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it takes off and moves forward while it rises to the specified altitude and turns to the specified heading. If a rotary-wing entity is in the air and is given this task with an altitude of zero, it moves forward and turns to the new heading while it descends to approximately 33 M and continues flying at that altitude on the new heading.

You can also quickly assign or change this task using the Entity Control Panel. For details, see "29.2. The Entity Control Panel" on page 567.

➤ **To assign a Fly Heading and Altitude task:**

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task > Movement > Fly Heading and Altitude**, or click the Fly Heading and Altitude button (). The Fly Heading and Altitude parameters display.

3. In the Heading box, specify the heading to fly to.
4. Next to the Heading box, indicate whether the entity should fly to the heading relative to true north or magnetic north.
5. In the Altitude box, specify the altitude above sea level to fly to.
6. In the Turn Rate box, specify the rate.
7. In the Climb/Descent Rate box, specify the rate.
8. Click OK.

30.12. Fly Heading

The Fly Heading task commands a fixed-wing or rotary-wing entity to turn to the specified heading at the specified turn rate. It maintains its current altitude and speed. When it is finished turning to the new heading, it continues to fly at this heading until it receives other commands. The entity will not exceed the turn rate specified in the OPD.

A fixed-wing entity that is on the ground ignores this task.

If a rotary-wing entity is on the ground and is given this task, it rises to approximately 25 meters above the terrain while simultaneously moving forward and turning to the specified heading. After turning to the new heading, it continues moving forward. Its altitude varies somewhat as it follows the terrain.

➤ **To assign a Fly Heading task:**

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task > Movement > Fly Heading**, or click the Fly Heading button (). The Fly Heading parameters display.
3. In the Heading box, specify the heading to fly to.
4. Next to the Heading box, indicate whether the entity should fly to the heading relative to true north or magnetic north.
5. In the Turn Rate box, specify the rate.
6. Click OK.

30.13. Fly Loiter Pattern

The Fly Loiter Pattern task commands an aircraft to fly in a loiter pattern, which can be a circle, figure eight, racetrack, cloverleaf, or a hold pattern. The entity loiters around the control point you specify. You also determine the turn direction, altitude, speed, and the duration of the loiter pattern flight. For the figure eight, racetrack, cloverleaf, and hold patterns, you specify the bearing and length to indicate the direction of the main axis and the space between loop centers.

The racetrack and hold patterns are both ovals. The approaches that an aircraft takes to enter them are more complicated than the other loiter patterns, and they also differ from each other.

- The hold pattern represents an air-traffic-control hold pattern, which requires the entity to fly a direct, parallel, or teardrop approach to that pattern.
- The racetrack pattern represents a typical military-style pattern. The aircraft flies along a tangent line to a point on either the inbound or outbound turn circle, then executes that turn to start the pattern.

The entity will not exceed the limits on altitude, air speed, or maximum lateral acceleration set in the OPD.

➤ **To assign a Fly Loiter Pattern task:**

1. Select a fixed-wing entity.
2. Choose **Task > Movement > Fly Loiter Pattern**. The Fly Loiter Pattern parameters display.
3. In the Loiter Point section, select the point tactical graphic that marks the reference point for the entity's loiter pattern. This is the center of every pattern except for the oval patterns: racetrack and hold. For those patterns, it is the point in the pattern where one of the turns begins.
4. In the Radius box, specify the radius of the loops made in the pattern.
5. Select the desired Turn Direction.
6. In the Altitude box, specify the altitude above sea level to fly the pattern.
7. In the Speed box, enter the speed to fly the pattern.
8. Specify the desired Duration Type:
   - Duration in Laps: Enter the number of laps the entity should fly the pattern. This must be a whole number.
   - Duration in Time: Enter the length of time that the entity should fly the pattern. The first box is the number of days, and the second box is hours, minutes, and seconds.
9. Specify the desired Loiter Pattern.
10. If the entity will fly in a figure 8, cloverleaf, racetrack, or hold pattern:
    a. In the Bearing box, enter the bearing of the main axis of the pattern.
    b. In the Length box, enter the space between loop centers.
11. Click OK.

30.14. Follow Entity

You can order an entity to follow a simulation object. You can specify a distance and offset between the simulation objects. If you tell two or more entities to follow the same simulation object, make sure that you do not give them the same offset (see "42.5.5 Following Simulation Objects" on page 945).

An entity continues to follow its target entity until:

- The task is overridden.
- The followed entity is destroyed.

If entity A is assigned to follow entity B, but it cannot find entity B, entity A continues to look for entity B until entity B becomes available, or until entity A is assigned another task.

The following entity tries to match the speed of the followed entity unless it has to hold back or catch up to accommodate changes in direction or terrain effects.

➤ **To assign a Follow task:**

1. Select the entity.
2. Choose **Task > Movement > Follow Entity**. The Follow Entity parameters display.
3. Optionally, filter the entity list.
4. In the Name box, type the name of the entity to follow, or select it from the list or on the terrain.
5. Optionally, type an offset in the Behind, Right, and Above text boxes. If you want an entity to follow in front, to the left, or below, put a minus sign (-) in front of the value.
6. Click OK.

30.14.1 How Fixed-Wing Entities Follow Entities

A fixed-wing entity can follow an entity that is on the ground, or might be on the ground, for example if it is following a fixed-wing entity that lands. In this case, to ensure that the follower does not crash, make sure that it is following above the followed entity and that the distance above is high enough to account for changes in the terrain.

30.15. Hover

Aircraft configured with the Fixed Wing VTOL movement system can perform a Hover task.

➤ **To assign a Hover task:**

1. Select the entity.
2. Choose **Task > Movement > Hover**. The Hover parameters display.
3. Select the location where the aircraft should land:
   - Click the location or enter the coordinates.
   - Select the waypoint.
4. Enter the height at which the entity should hover.
5. Click OK.

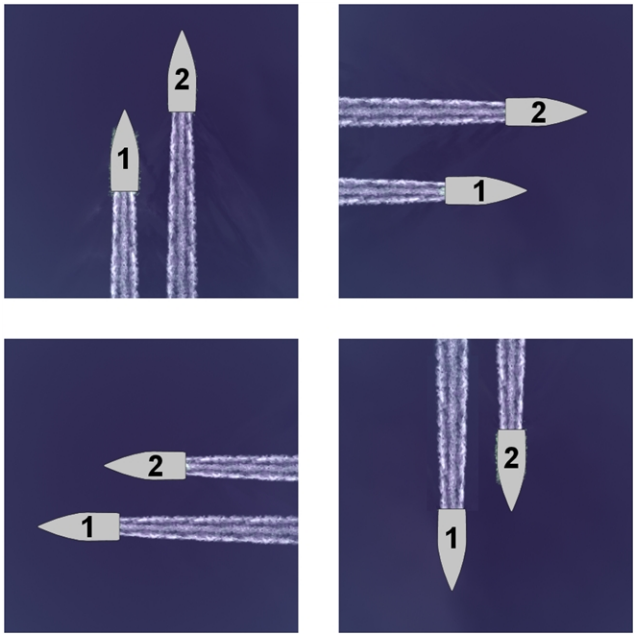
30.16. Keep Station

The Keep Station task directs a watercraft (surface or subsurface entity) to follow at a fixed distance and bearing from the leader's position, regardless of the leader's heading. If the station is located 200 yards to the northeast, the ship moves as needed to maintain that position, even if the leader changes the direction it is facing. For example, you can assign a destroyer to keep station between a carrier and the direction of probable enemy engagement, even when the task force changes course.

This task is different from the Follow Entity task, which positions the follower relative only to the leader's heading and causes the follower to adjust as needed when the leader changes direction.

Figure 229 illustrates a scenario where ship 1 is tasked to keep station with ship 2 at a 220° bearing relative to true north. This causes ship 1 to remain at a 220° offset from the center of ship 2, no matter what direction ship 2 is heading.

Figure 229. Keep Station example



The parameters you specify are:

- Name. The name of the entity to follow.
- Distance. Distance to maintain from the lead entity.
- Bearing. The bearing relative to the lead entity and either true or magnetic north where the watercraft should maintain its position.

➤ **To assign the Keep Station task:**

1. Select the entity that will be the follower.
2. Choose **Task > Movement > Keep Station**. The Keep Station parameters display.

3. Enter the parameters as desired.
4. Click OK.

30.17. Land At Airport

The Land at Airport task causes VR-Forces to land at specified aircraft on an appropriate runway based on weather conditions. This task requires use of a terrain database that has airport and runway data. For details, see "Chapter 67. Aerodromes" on page 1291.

➤ **To assign a Land At Airport task:**

1. Select the entity.
2. Choose **Task > Movement > Land At Airport**, or click the Land At Airport button (). The Land At Airport parameters display.
3. Select the aerodrome where you want the aircraft to land.
4. Click OK.

30.18. Land at Airport Near Location

The Land at Airport Near Location task causes VR-Forces to find an airport near the specified location and land the specified fixed-wing aircraft on an appropriate runway based on weather conditions. This task requires use of a terrain database that has airport and runway data. For details, see "Chapter 67. Aerodromes" on page 1291.

➤ **To assign a Land at Airport Near Location task:**

1. Select the fixed-wing entity.
2. Choose **Task > Movement > Land at Airport Near Location**, or click the Land At Airport button (). The Land At Airport Near Location parameters display.
3. Specify a range.
4. Specify the location in the dialog box or on the terrain. The list of aerodromes populates based on the location and range.
5. Select the aerodrome where you want the aircraft to land.
6. Click OK.

30.19. Land On Route

"24.1.6 How Fixed-Wing Entities Land" on page 512 describes how fixed-wing entities land and the steps to design a landing approach. The Land On Route task automates the process of landing a fixed-wing aircraft. However, you must specify a route to serve as the runway.

When the task is invoked, VR-Forces defines an approach route for the plane to move along. The length of this route is either the plane's altitude or the length needed to slow the plane down to landing speed, whichever is greater. The route is created and the plane moves along the route, coming to landing speed. At the end of the approach route, the plane is tasked to move along the runway.

Although it is possible to land an entity on another entity, such as an aircraft carrier, if you want to embark an entity, you must use the Embark task, not a landing task.

➤ **To assign a Land On Route task:**

1. Select the entity.
2. Choose **Task > Movement > Land On Route**. The Land On Route parameters display.
3. Select the route in the list or on the terrain, or enter the name of the route where you want it to land in the Route box.
4. Click OK.

30.20. Land On Runway

The Land On Runway task automates the process of landing a fixed-wing aircraft. There must be a runway in the terrain for the aircraft to land on, however; see "Chapter 67. Aerodromes" on page 1291. If the terrain does not have an aerodrome with runways that you want to use, you can add a route to the scenario and use that instead; see "30.19. Land On Route" on the previous page.

When the task is invoked, VR-Forces defines an approach route for the plane to move along. The length of this route is either the plane's altitude or the length needed to slow the plane down to landing speed, whichever is greater. The route is created and the plane moves along the route, coming to landing speed. At the end of the approach route, the plane is tasked to move along the runway.

- Although it is possible to land an entity on another entity, such as an aircraft carrier, if you want to embark an entity, you must use the Embark task, not a landing task.
- There is no logic to determine whether the runway is appropriate for the aircraft. For example, VR-Forces will not prevent a C-5 Galaxy from landing at a small municipal airport.

➤ **To assign a Land On Runway task:**

1. Select the entity.
2. Choose **Task > Movement > Land On Runway**, or click the Land On Runway button ( ). The Land On Runway parameters display.
3. In the Airport box, enter the ICAO code of the aerodrome. You can type the code, select a managed aerodrome in the Objects List, or select from the list of aerodromes at the bottom of the window. In the last case, any managed aerodromes are listed at the top of the list, and then all other aerodromes in the terrain are listed below those.
4. Select the specific runway on which the entity should land.
5. Click OK.

> If you want to design an explicit landing using a route, see "24.1.6 How Fixed-Wing Entities Land" on page 512, which describes how fixed-wing entities land and the steps to design a landing approach.

30.21. Land (Vertical)

Aircraft configured with the Fixed Wing VTOL movement system can perform a vertical landing task.

➤ **To assign a Land (Vertical) task:**

1. Select the entity.
2. Choose **Task > Movement > Land (Vertical)**. The Land (Vertical) parameters display.
3. Select the location where the aircraft should land:
   - Click the location or enter the coordinates.
   - Select the waypoint.
4. Click OK.

30.22. Maneuver Along

This task causes a ground vehicle unit to move along a route in a formation offset from the selected route.

The vehicle subordinates in the unit compute and traverse an offset route to the specified destination, making an attempt to remain together in some formation. Each subordinate computes an offset route and then traverses it by planning a path to each vertex in sequence at its own pace. There are a leader and followers, which use speed control to remain within some distance of each other as they move to the specified destination. If you change the ordered speed of the unit, it takes effect at the next movement, and speed control can override ordered speed.

➤ **To assign the Maneuver To task:**

1. Select the unit.
2. Choose **Task > Movement > Maneuver Along**. The Maneuver Along parameters display.
3. Select the route along which the ground unit should move.
4. Optionally, enter the speed at which the unit should maneuver to the location.
5. To have the unit start the route at the nearest vertex rather than moving to the beginning of the route, select the Start at Closest Vertex check box.
6. To have the unit move from the end of the route to the beginning, select Reverse Direction.
7. Click OK.

30.23. Maneuver To

This task causes a ground vehicle unit to move to a destination in a formation offset from each other. The subordinates in the unit move to their offset destinations and attempt to stay together. The offset destination is a subordinate position at the unit's destination, based on the current formation. The Maneuver To task always uses off-road navigation preferences.

The unit generates a two-vertex segment as a route. Each subordinate computes an offset route and then traverses it by planning a path to each vertex in sequence at its own pace. There are a leader and followers, which use speed control to remain within some distance of each other as they move to the specified destination. If you change the ordered speed of the unit, it takes effect at the next movement, and speed control can override ordered speed.

➤ **To assign the Maneuver To task:**

1. Select the unit.
2. Choose **Task > Movement > Maneuver To**. The Maneuver To parameters display.
3. Click Select Location or Select Object.
4. Specify the destination for the unit:
   - Click the location the unit should move to or enter the coordinates.
   - Select the simulation object.
5. Optionally, enter the speed at which the unit should maneuver to the location.
6. Click OK.

30.24. Move Along Route

By default, a simulation object moves along a route from its nearest vertex to its end point.

- In addition to routes, you can use other linear tactical graphics in this task, including corridors, lines, pedestrian paths, and supply routes.
- If you edit a route while a simulation object is moving to the starting point of that route as part of a Move Along Route task, the simulation object moves to the nearest vertex, whether or not it is the starting point.
- If a human is moving along a route outside of a navigation area and you move it away from the route or change the route, the entity plans a path to the route vertex it is moving to rather than moving to the nearest point on the route and then proceeding to the target vertex.
- If you want a simulation object to repeatedly move along the same route from beginning to end (for example, to follow a circular route), be sure to clear the Start at Closest Vertex check box. If you do not, after the first time the simulation object moves along the route, it keeps going to the end vertex because that is the closest vertex to where it is starting the new Move Along Route task.

- When you task an air platform entity to follow a route, be sure to take into account the altitude of the route's vertices and the terrain the entity has to pass over to complete the task. (For more information, see "24.1.4 Fixed-Wing Entity Movement on the Ground and in the Air" on page 511 and "24.2. Rotary-Wing Entity Behaviors" on page 513.)

When a unit performs a Move Along Route task:

- Subordinates in the unit may use the Maneuver Along subtask to move along the route, for example, individual vehicles or ground vehicle subunits that are configured to use Maneuver Along. See "30.22. Maneuver Along" on page 598 to learn more about that task. (Ground vehicle units are equipped with maneuver tasks by *ground-disaggregated-movement.sysdef*.)
- Subordinates that are not equipped with ground vehicle maneuver tasks, such as human entities, execute the move-along task as described above. When you assign the Move Along Route task to a unit comprised of both ground vehicles and humans, the entities perform the task in potentially different ways that may involve path planning or dynamic obstacle avoidance.

➤ **To assign a Move Along Route task:**

1. Select the simulation object.
2. Choose **Task > Movement > Move Along Route**, or click the Move Along Route button (). The Move Along Route parameters display.
3. In the Name box, enter the route you want the simulation object to follow.
4. To have the simulation object move from the end of the route to the beginning, select Reverse Direction.
5. To have a ground vehicle use the road driving movement strategy, select the Treat Route as Road check box. The route is treated as a two-lane bidirectional road. Ground vehicles compute curves to go around the vertices of the route. (This applies only to entity-level scenarios.)

   ⚠ If you select Treat Route as Road, the entity must be within 10 meters of the route or the task will fail.

6. To have the simulation object start the route at the nearest vertex rather than moving to the beginning of the route, select the Start at Closest Vertex check box.
7. Click OK.

30.25. Move Along Route with Actions

Use Move Along Route with Actions task to assign actions to an entity that is following a route. The tasked entity performs the actions at points in the route that you specify, then continues to follow the route.

- By default, a simulation object executing a Move Along Route with Actions task goes to the beginning of the route, rather than starting the route at its nearest vertex.
- If a human entity is moving along a route outside of a navigation area and you move it away from the route or change the route, the entity plans a path to the route vertex it is moving to rather than moving to the nearest point on the route and then proceeding to the target vertex.
- If you want a simulation object to repeatedly move along the same route from beginning to end (for example, to follow a circular route), be sure to clear the Start at Closest Vertex check box. If you do not, after the first time the simulation object moves along the route, it keeps going to the end vertex, because that is the closest vertex to where it is starting the new Move Along Route with Actions task.
- The Move Along Route with Actions task includes a Specify Time on Target option where you can set the time when the tasked entity must reach the target location (TL) on the route. When you set a time on target, the task employs a hold location (HL) where the tasked entity can wait or circle to use up extra time and complete the task at the specified time.
- You can control the tasked entity's speed by setting Ordered Speed, which offers Enforced and Suggested options. However, if you set both a time on target and an enforced ordered speed, the entity might not be able to complete the task.
- The Abandon if Time on Target Cannot Be Achieved option is available when you use Specify Time on Target. If you select the option, the tasked entity abandons the task if it determines that it cannot reach the target location within 10 seconds of the specified time.
- When you task an aircraft entity to follow a route, take into account the altitude of the route's vertices and the terrain the entity must pass over to complete the task. If you are using an existing ground route, you can set an ordered altitude that puts the tasked entity at the height you want, or raise the vertices to enable the entity to follow the route. (For more information, see "24.1.4 Fixed-Wing Entity Movement on the Ground and in the Air" on page 511 and "24.2. Rotary-Wing Entity Behaviors" on page 513.)
- For aircraft, the flight path shown in the terrain profile in the Move Along Route with Actions dialog box is an approximation, not a precise rendering.
- In addition to routes, you can use other linear tactical graphics in this task, including air axes, air corridors, lines, and supply routes.

➤ **To prepare to assign the task:**

1. Identify the entity that will follow the route, or use the Create Object Palette to create one.

   ⓘ If you create an airplane in the air, set its altitude to at least a few thousand feet to ensure it doesn't stall.

2. Identify the route (or other linear tactical graphic) the entity will follow, or use the Create Object Palette to create one.

3. If any task you will use relies on tactical graphics such as target points or areas, use the Create Object Palette to create them along the route.

> To drop a bomb, you must create a target point on the route. You cannot drop a bomb on the route's target location, which is simply the end point of the route.

➤ **To assign a Move Along Route with Actions task:**

1. Select the entity that will move along the route and perform the actions.
2. Choose **Task > Movement > Move Along Route with Actions**, or click the Move Along Route With Actions button (). The Move Along Route with Actions parameters display.
3. In the Name box, enter the route or other linear tactical graphic that the entity should follow. The dialog box displays a terrain profile of the route you selected and a numbered list of the route's vertices.
4. Right-click any of these to add an action:
   - In the Move Along Route with Actions dialog box, any location on the route in the terrain profile.
   - In the Action list, any vertex.
   - In the window, any location on the route.
5. In the context menu, choose **Insert Task** or **Insert Set** and select an option. For example, you might choose **Insert Task > Engagement > Release Bomb on Target**or **Insert Set > Action > Ordered Altitude**. The task or set parameters display.
6. Enter the necessary information in the parameters and then click OK.

   The action is added to the Action list in the Move Along Route with Actions dialog box.
7. Optionally, set a speed for the entity:
   a. Right-click a point on the route and then select **Insert Set > Action > Ordered Speed**.
   b. In the Ordered Speed parameters, enter a speed. For aircraft entities, this is the ground speed.
   c. Optionally, select Enforced. (If you also Specify Time on Target below, the entity might not be able to achieve the time on target at the ordered speed unless you change or remove the ordered speed later in the task. You can do that by adding another **Insert Set > Action > Ordered Speed** for a different speed, or even for the same speed by selecting Suggested rather than Enforced.)
   d. Click OK. VR-Forces updates the estimated times in the Move Along Routes with Actions dialog box.
8. Optionally, select Specify Time on Target and then set the time at which you want the simulation object to reach the target point in the route. When you enable Specify Time on Target, VR-Forces recalculates the estimated times in the Action

list, counting backward from the Time on Target at the route's target location. (Estimated times before the hold location do not change.) When it is not possible for the tasked entity to achieve an action in the time allowed, the numbers in the Time on Target field and the estimates for the actions in question turn red.

9. If you select Specify Time on Target, you can select Abandon if Time on Target Cannot Be Achieved. This tells the tasked entity to give up on the task if it estimates that it cannot complete the task by the time you specified in Time on Target (within 10 seconds).
10. To have the simulation object move from the end of the route to the beginning, click Reverse Route.
11. To reverse the order of the actions along the route, click Reverse Actions. You can also reorder by right-clicking an action or the target location in the Actions list and selecting Move. You can then click another point for the entity to perform that action.
12. To have the simulation object start the route at the nearest vertex rather than moving to the beginning of the route, select the Start at Closest Vertex check box.
13. Click OK.

30.26. Move Into Formation

You can order a ground unit to move to a formation at a specified location. The members of the unit to travel through the terrain to the specified location and into the new formation. If you want the unit to assume a different formation but not travel to another location, see "30.52. Transition Into Formation" on page 621.

Moving to a formation is not the same as setting formation. Setting formation changes the formation instantly, as described in "40.33. Formation" on page 879.

➤ **To assign a Move Into Formation task:**

1. Select the unit.
2. Choose **Task > Movement > Move Into Formation**. The Move Into Formation parameters display.
3. Select the desired formation from the list.
4. Specify the location at which the unit is to move into formation by entering coordinates, or by clicking on the terrain.
5. Specify the heading the unit is to assume at the target location.
6. Click OK.

30.27. Move Next to Target

The Move Next to Target task commands an entity to move to a position beside the specified simulation object, which can be either another entity or a point tactical graphic. It moves to the side of the object that it is closest to when it begins the task, whether that be the front, back, left or right side.

➤ **To assign a Move Next to Target task:**

1. Select the entity.
2. Choose **Task > Movement > Move Next to Target**. The Move Next to Target parameters display.
3. Select the Target Object.
4. Click OK.

30.28. Move To

You can order an entity to move to a location or point. If navigation data is present, simulation objects take it into account in planning their path to the location.

Moving to a location or point is not the same as setting location with a set data request. Setting location moves a simulation object instantaneously. Moving to a location or point causes a simulation object to travel through the terrain.

If you want a ground platform entity to use the road network to move to a location, first use the Navigation Preferences set data request to ensure that it is set to Prefer Roads. For details about road movement, see "23.2.4 Vehicle Movement On Roads" on page 503.

For certain simulation objects, if you add a Move To task to a plan, you can select the Continue On check box to cause the simulation object continue moving to the next destination. This option does not apply when you assign the task as an independent one. This is only available to rotary-wing aircraft, watercraft, and humans. When the option is selected for a rotary-wing aircraft, watercraft but there are no further specified destinations or other movement tasks, the entity slows to a halt; humans simply stop moving.

When a unit performs a Move To task:

- Each subordinate plans and traverses its own path to an "offset destination", without consideration of the others. The offset destination is a subordinate position at the unit's destination, based on the current formation.
- The unit's orientation at the end of the movement faces toward the task's destination, away from the starting position.

ⓘ Rotary-wing entities are not able to reach moving waypoints or land on moving ships. A rotary-wing entity will approach a given waypoint or landing point, but will never completely reach the goal.

➤ **To assign a Move To task:**

1. Select the simulation object.
2. Choose **Task > Movement > Move To**, or click the Move To button (). The Move To parameters display.
3. Optionally, click Select Location or Select Object.
4. Select the destination:

- Click the location to move to or enter the coordinates.
- Select the waypoint or point tactical graphic.

5. For a rotary- or fixed-wing aircraft moving to a location, you can set the altitude, as described in "14.4. Specifying an Object's Altitude" on page 386.

> If the destination is a location and the task is for an aircraft that is in the air, the default is for it to remain at the same altitude. If there are obstacles in the terrain, such as buildings, you might want to change the altitude to a height that will prevent the entity from crashing into them.

6. If you are adding the task to a plan and you want the simulation object to continue to the next location, select Continue On.
7. Click OK.

30.29. Move to Altitude

You can order an aircraft entity to move to an altitude. When you assign this task, the entity moves to the new altitude at the same coordinates. Fixed-wing aircraft spiral to the new coordinates. Rotary-wing entities rise or descend.

Moving to an altitude is not the same as setting altitude. Setting altitude moves an entity instantaneously. It is also not the same thing as Fly Altitude. At the conclusion of Move to Altitude, an aircraft circles or hovers at the new altitude. At the conclusion of Fly Altitude, an aircraft continues moving at its current heading.

> Fixed-wing entities that are on the ground cannot respond to a Move to Altitude task. If they are given this task, a message prints to the object console and the entity ignores the task.

➤ **To assign a Move to Altitude task:**

1. Select the entity.
2. Choose **Task > Movement > Move to Altitude**, or click the Move to Altitude button (). The Move to Altitude parameters display.
3. Enter the altitude that you want the entity to move to.
4. Select whether it is above mean sea level (MSL) or ground level (AGL).
5. Click OK.

30.30. Move to Bullseye Offset

Use the Move to Bullseye Offset task to send an aircraft to a bearing and range relative to a bullseye tactical graphic in the scenario.

➤ **To assign a Move to Bullseye Offset task:**

1. Select the entity.
2. Choose **Task > Movement > Move to Bullseye Offset**. The Move to Bullseye Offset parameters display.
3. Select the bullseye.
4. Enter the bearing from the bullseye to the location where you want the aircraft to move to. Indicate whether the heading is relative to true or magnetic north.
5. Enter the range.
6. Click OK.

30.31. Move to Depth

The Move to Depth task lets you specify the depth for subsurface entities. When an entity moves to a depth, it continues any horizontal movement that was in effect prior to starting the task.

You can also quickly assign or change this task using the Entity Control Panel. For details, see "29.2. The Entity Control Panel" on page 567.

➤ **To assign a Move to Depth task:**

1. Select the entity.
2. Choose **Task > Movement > Move to Depth**. The Move to Depth parameters display.
3. Select an option for the depth. If you choose a specific depth, type a depth in the Target Depth box.
4. Specify the ascent/descent rate.
5. Click OK.

30.32. Move To (Plan Path)

The Move to (Plan Path) task applies to surface and subsurface watercraft entities. It plots a route to the waypoint based on the depth of water between the starting point and ending point. You specify the minimum depth required for the ship to maneuver. You can specify that the depth be based on the ship's draft or you can enter a specific depth value. The ship's draft is determined by calculating the distance its bounding box extends below its origin, as set in the Simulation Object Editor.

If you give this task to a submarine, VR-Forces assumes that it is moving on the surface.

- For this task to work, your terrain must include S-57 data or depth information provided in a similar way.
- The formula for calculating the draft is:

  $$draft = height/2 - offset(Up)$$

  where *height* and *offset* are set in the Size group box in the Simulation Object Editor.

➤ **To assign a Move to (Plan Path) task:**

1. Select the entity.
2. Choose **Task > Movement > Move to (Plan Path)**. The Move to (Plan Path) parameters display.
3. Select the destination:
   — Click the location to move to or enter the coordinates.
   — Select the waypoint or point tactical graphic.
4. Specify how you want VR-Forces to determine the minimum depth of water for the route.
5. If you want to see the path displayed, select Display Route.
6. Click OK.

30.33. Orbit

The Orbit task allows an aircraft to circle around a point without needing to create a route for it to follow.

➤ **To assign an Orbit task:**

1. Select the entity that you want to fly the orbit.
2. Choose **Task > Movement > Orbit**. The Orbit parameters display.
3. Specify the coordinates of a point on the terrain around which the entity should orbit. You can click on the terrain to specify the point. You can also expand the location group box by clicking the arrow next to the coordinates and altitude, then enter the location for the entity to orbit.
4. Specify the altitude at which to fly.
5. Specify the radius of the circle.
6. Select or clear the Clockwise option to specify the direction of travel.
7. Click OK.

30.34. Orbit Object

The Orbit Object task causes an aircraft to fly a circular pattern around a simulation object or point tactical graphic. If the object moves, the aircraft moves to maintain the orbital relationship.

➤ **To assign an Orbit Object Task:**

1. Select the entity that you want to orbit.
2. Choose **Task > Movement > Orbit Object**, or click the Orbit Object button (). The Orbit Object parameters display.
3. Select the simulation object or tactical graphic around which you want the entity to orbit.
4. In the Orbit Radius box, specify the radius of the orbit.
5. To orbit clockwise, select the Clockwise check box.
6. Click OK.

30.35. Paratrooper Air Drop

The Paratrooper Air Drop task causes an aircraft to disembark the selected paratrooper entities and cause them to parachute toward a target object or location. The specified paratroopers try to land as close to the target as they can given the altitude at which they disembarks and the distance to the target. After landing, each paratrooper walks to the muster point.

The aircraft performing the Airdrop task must have the Air Drop system (*airDrop.sysdef*). The embarked entities must be Paratroopers.

- While an entity is parachuting, if you double-click that entity to try to move it, it snaps to the terrain.
- You cannot specify an order for the paratroopers. They drop in alphanumerical order. If you want them to drop in a particular order, see "34.1.4 Airdrop" on page 683 and "34.1.5 Airdrop from Aircraft" on page 684.

➤ **To assign the Paratrooper Air Drop task:**

1. Select the aircraft that the paratrooper entities are embarked upon.
2. Choose **Task > Movement > Paratrooper Air Drop**. The Paratrooper Air Drop parameters display.
3. Specify a target point where the paratroopers will try to land.
4. Specify a target point where the paratroopers should move after they land.
5. Select each of the paratroopers embarked on the aircraft that should parachute to the ground.
6. You can edit the number of seconds between the paratroopers drops.
7. You can edit the altitude at which to deploy the parachutes. If the aircraft is below the deployment altitude, the parachutes deploy as soon as the entities disembark.
8. By default, Deploy Altitude is AGL is not selected and the deployment altitude is above mean sea level (MSL). Select the check box to use AGL instead.

9. If the aircraft is a rotary wing, you can specify a drop route for it to follow as it air drops the paratroopers.
10. Click OK.

30.36. Patrol Area

The Patrol Area task is for watercraft (surface and subsurface entities). This allows you to have a ship patrol an area such as a harbor. The task uses a line between the two most distant points that falls within the area and then moves the ship in a zig-zag pattern across the area with that line as the spine of the pattern.

The ship can be outside of the area at the beginning of the task. It will move into the area to patrol it.

➤ **To assign a Patrol Area task:**

1. Select the entity.
2. Choose **Task > Movement > Patrol Area**. The Patrol Area parameters display.
3. Select the area for the ship to patrol.
4. Click OK.

30.37. Patrol Between

An entity can patrol between two point objects, simulation objects, or both. When a simulation object patrols between two objects, it goes to the starting object and then moves back and forth between the two objects.

➤ **To assign a Patrol Between task:**

1. Select the entity.
2. Choose **Task > Movement > Patrol Between**, or click the Patrol Between button (). The Patrol Between parameters display.
3. In the Name box for Point 1, type the name of the first object to patrol between, or select an object in the list or on the terrain.
4. In the Name box for Point 2, type the name of the second object to patrol between, or select an object in the list or on the terrain.
5. Click OK.

30.38. Patrol Route

When an entity patrols along a route, it moves to the nearest vertex of the route, to the end point, back to the beginning of the route, and then continues going back and forth along the route until given another command.

If a human is moving along a route outside of a navigation area and you move it away from the route or change the route, the entity plans a path to the route vertex it is moving to rather than moving to the nearest point on the route and then proceeding to the target vertex.

> For more information about moving along routes, see the issues discussed in "30.24. Move Along Route" on page 599.

➤ **To assign a Patrol Route task:**

1. Select the simulation object.
2. Choose **Task > Movement > Patrol Route**, or click the Patrol Route button (). The Patrol Along Route parameters display.
3. In the Name box, type the name of the route to patrol, or select the route in the list or on the terrain.
4. Click OK.

30.39. Pattern Hold

The Pattern Hold task causes a fixed-wing or rotary-wing entity to execute a racetrack hold pattern at the specified location or waypoint.

In a standard racetrack, the aircraft tries to spend one minute per leg. Therefore, the length of the leg varies with the speed. The default turn radius is three degrees/second. In a custom racetrack, you specify the length and width of the legs.

The aircraft enters the pattern using direct, parallel, or teardrop entry. It determines the approach based on its location relative to the desired pattern. If you set the aircraft's object console to Info, the chosen entry method is displayed.

It may take an aircraft one or two loops before it settles into the desired pattern.

➤ **To assign a Pattern Hold task:**

1. Select a fixed-wing or rotary-wing entity.
2. Choose **Task > Movement > Pattern Hold**, or click the Pattern Hold button (). The Pattern Hold parameters display.
3. Using the Holding Fix section, select the location or waypoint.
4. Specify the heading, speed, and altitude for the hold pattern.
5. Specify the turn direction for the hold pattern.
6. For the Pattern Type, select Time or Distance. If you select Time, specify the amount of time per leg of the pattern. If you select Distance, specify a length and width.
7. Click OK.

30.40. Refuel From

The Refuel From task causes an aircraft with the Air Refueling Receiver system to rendezvous with an air tanker that is performing the Establish Refueling Station task. See "30.7. Establish Refueling Station" on page 587 for more information.

➤ **To assign a Refuel From task:**

1. Select a fixed-wing entity.
2. Choose **Task > Movement > Refuel From**. The Refuel From parameters display.
3. Select the air tanker that the aircraft should refuel from.
4. Click OK.

30.41. Return To Base

You can quickly task an entity to return to its assigned base aerodrome (RTB). This task requires that you assign a base aerodrome using the Base set data request, as described in "40.15. Base" on page 872, or the Edit dialog box for the aircraft. If it is a managed aerodrome, at least one of its runways must be set to active to allow it to land. See "67.3.4 Runway Active" on page 1297.

When you task an entity to RTB, it immediately plans a route back to its base. It does not change altitude until it is 10 NM from the base, at which point it makes a landing. The entity is not removed from the scenario after it lands on a runway at its base.

➤ **To task an aircraft to return to base:**

1. Select the entity.
2. Choose **Task > Movement > Return to Base**, or click the Return to Base button (). The entity returns to its assigned air base.

30.42. Rotary Wing Airlift Cargo to Location

The Rotary Wing Airlift Cargo to Location task causes an unmanned rotary-wing aircraft to move to a list of objects, embark them onto itself, move to a location, and disembark the objects. The objects must be configured as embarkable and the unmanned aircraft must have the Quadcopter or Quadcopter Heavy Lift movement dynamics system.

If the total weight of the objects plus the weight of the aircraft is greater than the aircraft can carry, the aircraft will embark the objects, but be unable to fly.

➤ **To assign a Rotary Wing Airlift Cargo to Location task:**

1. Select an unmanned aircraft.
2. Choose **Task > Movement > Rotary Wing Airlift Cargo to Location**. The Rotary Wing Airlift Cargo To Location parameters display.
3. Optionally, filter the list of object types.
4. Type the name of an object in the Name box, or in the Cargo Objects list select one or more objects.
5. In the Cruise Altitude box, specify the altitude at which you want the aircraft to fly.
6. Specify the destination location by typing in the coordinates, or click the Destination Location filter button () to give the Destination Location focus, and click on the terrain to specify the destination location.
7. Click OK.

30.43. Rotary Wing Land

The Rotary Wing Land task automates the process of landing a rotary-wing aircraft. Rotary-wing aircraft can land on points.

> ⚠ Although it is possible to land an entity on another entity, such as an aircraft carrier, if you want to embark an entity, you must use the Embark task, not a landing task.

➤ **To assign a Rotary Wing Land task:**

1. Select the entity.
2. Choose **Task > Movement > Rotary Wing Land**. The Rotary Wing Land parameters display.
3. In the Name box, select the point in the list or on the terrain where you want the entity to land, or type the name of the waypoint.
4. Click OK.

30.44. Rotary Wing Retrieve Cargo

The Rotary Wing Retrieve Cargo task causes an unmanned aircraft to move to one or more cargo items and embark them onto itself. It hovers at the location of the last item retrieved until given further tasks.

➤ **To assign a Rotary Wing Retrieve Cargo task:**

1. Select an unmanned aircraft.
2. Choose **Task > Movement > Rotary Wing Retrieve Cargo**. The Rotary Wing Airlift Cargo To Location parameters display.
3. Optionally, filter the list of object types.
4. Type the name of an object in the Name box or in the Cargo Objects list, select one or more objects.
5. In the Cruise Altitude box, specify the altitude at which you want the aircraft to fly.
6. Click OK.

30.45. Rotary Wing Unload Cargo

The Rotary Wing Unload Cargo task causes a rotary-wing aircraft to unload (disembark) its cargo at the current location of the aircraft.

➤ **To assign a Rotary Wing Unload Cargo task:**

1. Select an unmanned aircraft.
2. Choose **Task > Movement > Rotary Wing Unload Cargo**. The aircraft unloads its cargo.

30.46. Sail Heading

The Sail Heading task causes a surface or subsurface entity to sail towards the specified heading. After turning to the heading, the entity moves at its ordered speed indefinitely.

> You can also quickly assign or change this task using the Entity Control Panel. For details, see "29.2. The Entity Control Panel" on page 567.

➤ **To assign a Sail Heading task:**

1. Select the entity.
2. Choose **Task > Movement > Sail Heading,** or click the Sail Heading button (). The Sail Heading parameters display.
3. Specify a heading in the indicated units.
4. Next to the Final Heading box, indicate whether the entity should sail to the heading relative to true north or magnetic north.
5. Specify the diameter of the turn.
6. Specify the direction of the turn:
   - Toward Heading. Chooses the most direct turn.
   - Port. Turns to port until it reaches the heading.
   - Starboard. Turns to starboard until it reaches the heading.
7. Click OK.

30.47. Search and Rescue

There are three search and rescue patterns available to aircraft: creeping line, expanding square, and parallel.

30.47.1 Fly Creeping Line Pattern

The Fly Creeping Line Pattern search and rescue task causes an aircraft to fly a pattern starting at the Commence Search Point. You specify the Major Axis for the direction in which the search expands (creeps). The search legs are perpendicular to this axis. The Search Area Length is the size of the search area (box) in the direction of the Major Axis, and the Search Area Width is the size perpendicular to the Major Axis. The search area is either to the right or the left of the Commence Search Point.

The entity will not exceed the limits on altitude and air speed set in the OPD.

Figure 230. Fly Creeping Line Pattern dialog box

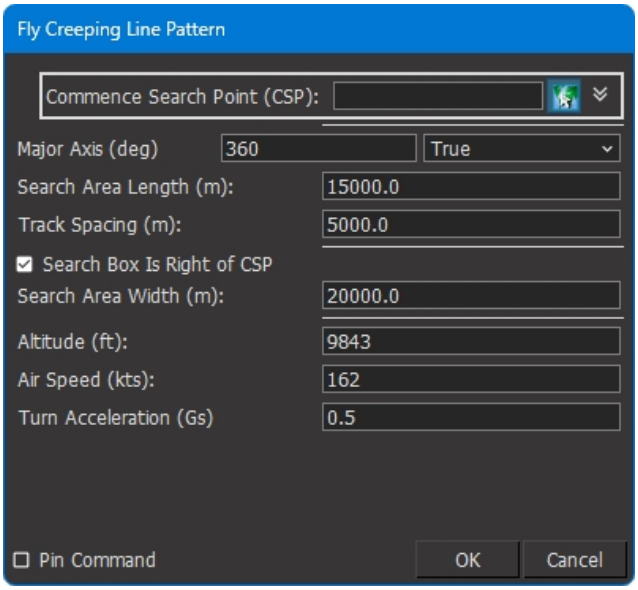
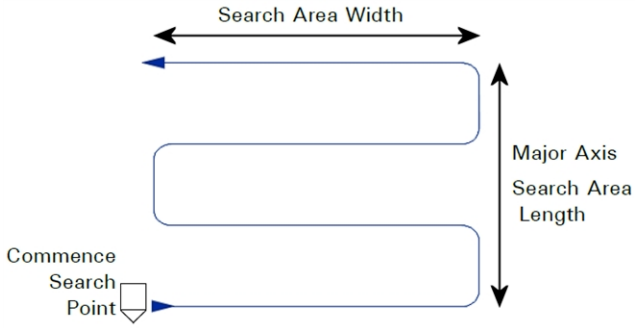


Figure 231. Creeping Line Pattern, search right of CSP



➤ **To assign a Fly Creeping Line Pattern search and rescue task:**

1. Select an aircraft.
2. Choose **Task > Movement > Search and Rescue > Fly Creeping Line Pattern**. The Fly Creeping Line Pattern parameters display (Figure 230).
3. Specify the parameters:
   - In the Commence Search Point section, select the waypoint.
   - In the Major Axis box, enter the axis for the search. Next to this box, indicate whether the entity should fly relative to true north or magnetic north.
   - In the Search Area Length box, enter the length of the search area. This is measured along the Major Axis.
   - In the Track Spacing box, enter the spacing between the search legs.

- Select the Search Box Is Right of CSP check box to search the area to the right of the Commence Search Point. Clear the check box to search the area to the left of the waypoint.
- In the Search Area Width dialog box, enter the width of the search area. This is the width measured from the Commence Search Point.
- In the Altitude box, specify the altitude above sea level to fly the search and rescue pattern.
- In the Air Speed box, enter the speed to fly the pattern.
- In the Turn Acceleration box, enter the maximum Gs for the turns. This acceleration, together with the selected air speed, determines the radius of the loops in the pattern. You must set the air speed and turn acceleration such that the aircraft can turn 180 degrees within the track spacing distance.

4. Click OK.

30.47.2 Fly Expanding Square Pattern

The Fly Expanding Square Pattern search and rescue task causes an aircraft to fly a pattern that starts at the Commence Search Point. That point is the center of the search, and the entity moves around it in an expanding series of right turns until the search box is covered.

The entity will not exceed the limits on altitude and air speed set in the OPD.

Figure 232. Fly Expanding Square Pattern dialog box

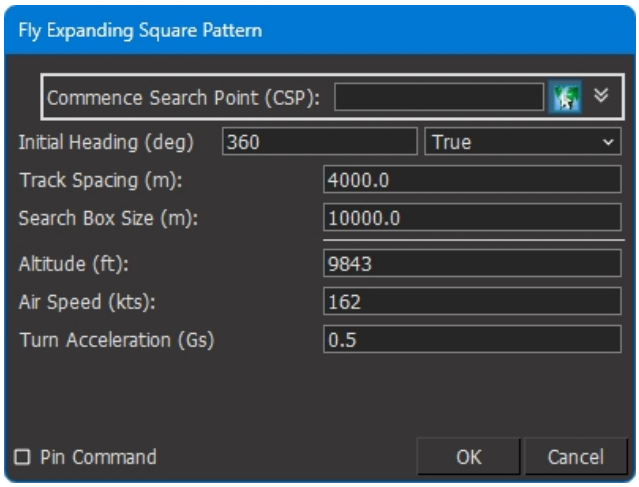
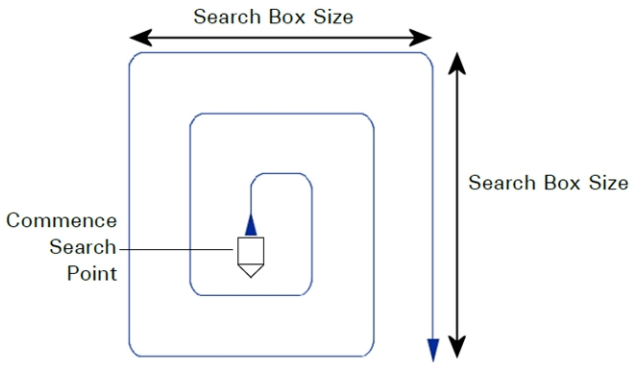


Figure 233. Expanding Square Pattern



➤ **To assign a Fly Expanding Square Pattern search and rescue task:**

1. Select an aircraft.
2. Choose **Task > Movement > Search and Rescue > Fly Expanding Square Pattern**. The Fly Expanding Square Pattern parameters display (Figure 232).
3. Specify the parameters:
   - In the Commence Search Point section, select the waypoint.
   - In the Initial Heading box, enter the heading at which the entity should begin the pattern. Next to this box, indicate whether the entity should fly to that heading relative to true north or magnetic north.
   - In the Track Spacing box, enter the spacing between the search legs.
   - In the Search Box Size box, enter the length for the sides of the search square.
   - In the Altitude box, specify the altitude above sea level to fly the search and rescue pattern.
   - In the Air Speed box, enter the speed to fly the pattern.
   - In the Turn Acceleration box, enter the maximum Gs for the turns. This acceleration, together with the selected air speed, determines the radius of the loops in the pattern. You must specify an air speed and turn acceleration such that the aircraft can turn 180 degrees within the track spacing distance (the first two turns).
4. Click OK.

30.47.3 Fly Parallel Pattern

The Fly Parallel Pattern search and rescue task causes a aircraft to fly a pattern starting at the Commence Search Point. You specify a Major Axis. The search legs are in line with this axis, and the search area expands perpendicularly to the distance of the Search Area Width. The Search Area Length is the size of the search area (box) in the direction of the Major Axis. The search area is either to the right or the left of the Commence Search Point.

The entity will not exceed the limits on altitude and air speed set in the OPD.

Figure 234. Fly Parallel Pattern dialog box

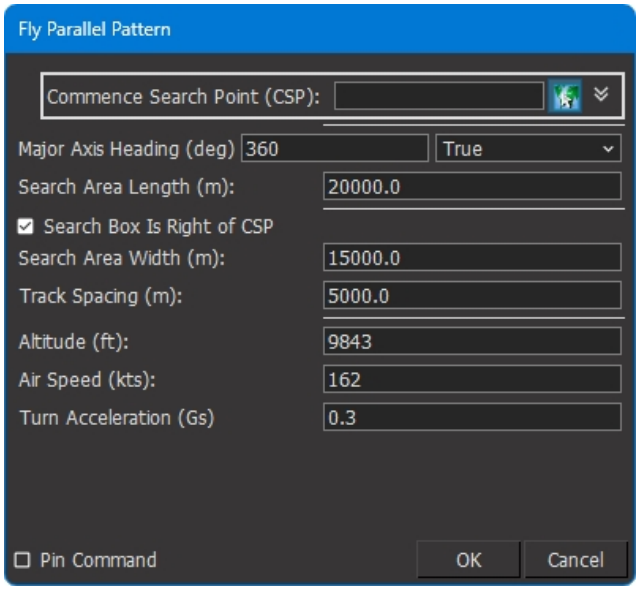
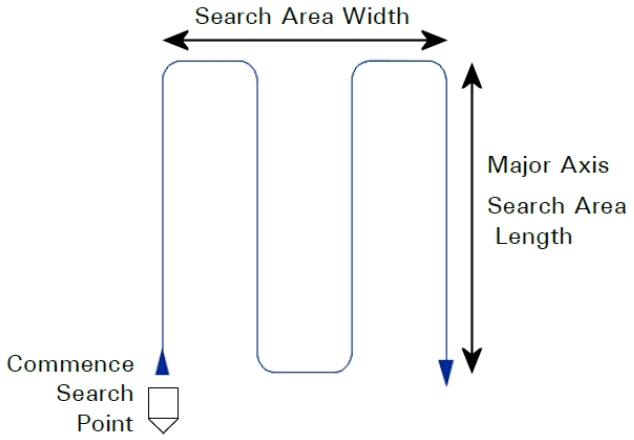


Figure 235. Parallel Pattern, search right of CSP



➤ **To assign a Fly Parallel Pattern search and rescue task:**

1. Select a aircraft.
2. Choose **Task > Movement > Search and Rescue > Fly Parallel Pattern**. The Fly Parallel Pattern parameters display(Figure 234).
3. Specify the parameters:
   - In the Commence Search Point section, select the waypoint.
   - In the Major Axis Heading box, enter the axis for the search. Next to the this box, indicate whether the entity should fly relative to true north or magnetic north.
   - In the Search Area Length box, enter the length of the search area. This is measured along the Major Axis.

- In the Track Spacing box, enter the spacing between the search legs.
- Select the Search Box Is Right of CSP check box to search the area to the right of the Commence Search Point. Clear the check box to search the area to the left of the waypoint.
- In the Search Area Width dialog box, enter the width of the search area. This is the width measured from the Commence Search Point.
- In the Altitude box, specify the altitude above sea level to fly the search and rescue pattern.
- In the Air Speed box, enter the speed to fly the pattern.
- In the Turn Acceleration box, enter the maximum Gs for the turns. This acceleration, together with the selected air speed, determines the radius of the loops in the pattern. You must specify an air speed and turn acceleration such that the aircraft can turn 180 degrees within the track spacing distance.

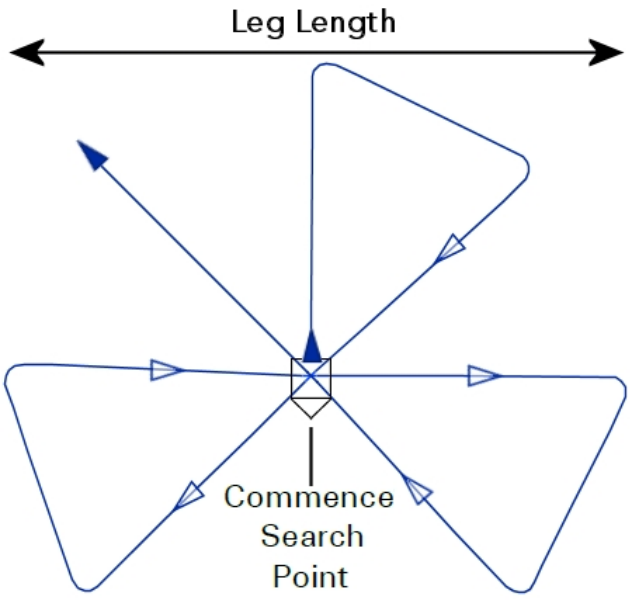
4. Click OK.

30.47.4 Sector Search

The Sector Search pattern search and rescue task causes an aircraft to fly a pattern designed for searching a small area.

The entity will not exceed the limits on altitude, air speed, and turning radius set in the OPD.

Figure 236. Sector Search pattern



➤ **To assign a Sector Search pattern search and rescue task:**

1. Select a aircraft.
2. Choose **Task > Movement > Search and Rescue > Sector Search**. The Sector Search parameters display.
3. Specify the parameters:
   - In the Commence Search Point section, click a location where the search should begin or enter the coordinates.
   - In the Leg box, enter the length of each leg of the search. This is the distance from one point where the aircraft will turn to the next (see Figure 236 on the previous page).
   - In the Altitude box, specify the altitude above sea level to fly the search and rescue pattern.
   - In the Air Speed box, enter the speed to fly the pattern.
4. Click OK.

30.48. Sweep Naval Mines

The Sweep Naval Mines task causes an entity to look for and disable naval mines along a route. It looks within a specified distance from the route. An error rate specifies a percentage chance that the entity will detect a mine, but not disable it. The Sweep Naval Mines task is available to surface entities that have a Naval Mine Sweep system.

➤ **To assign a Sweep Naval Mines task:**

1. Select the entity.
2. Choose **Task > Movement > Sweep Naval Mines**. The Sweep Naval Mines parameters display.
3. Select the route to follow.
4. Specify the sensing range. The default is 300 meters.
5. Optionally, specify the disarm failure probability. This is the percentage chance that a mine will be found, but not disabled.
6. Click OK.

30.49. Takeoff On Route

You can explicitly task a fixed-wing entity to take off. (For details about implicit take-off, see "24.1.5 How Fixed-Wing Entities Take Off" on page 511.) In a Takeoff On Route task, the entity moves along the specified route, reaches takeoff speed (30% of minimum lift), and then moves to a point defined by the controller some distance up and away from the end of the route.

➤ **To assign a Takeoff On Route task:**

1. Select the entity.
2. Choose **Task > Movement > Takeoff On Route**. The Takeoff On Route parameters display.
3. Select the route.
4. Optionally, select the Teleport to Runway Start check box to have the entity move immediately to the start of the runway. This saves some time by quickly positioning the entity for takeoff.
5. Click OK.

30.50. Takeoff On Runway

In a Takeoff on Runway task, the aircraft moves along either the specified runway or the closest one, reaches takeoff speed (30% of minimum lift), and then moves to a point defined by the controller some distance up and away from the end of the runway. If the entity is not at the end of the runway, and you do not choose to have it teleport to the runway or create a route for it to follow to the end of the runway, then it moves straight across the terrain to the runway's starting point. It does not move along the taxiways.

This task requires a terrain where there are aerodromes with runways for the aircraft to take off from; see "Chapter 67. Aerodromes" on page 1291. If the terrain does not have an aerodrome with runways that you want to use, you can add a route to the scenario and use that instead; see "30.49. Takeoff On Route" on the previous page.

ⓘ There is no logic to determine whether the runway is appropriate for the aircraft. For example, VR-Forces will not prevent a B-52 Stratofortress from taking off on a small municipal airport's runway.

➤ **To assign a Takeoff On Runway task for a specific runway:**

1. Select the entity.
2. Choose **Task > Movement > Takeoff On Runway**, or click the Takeoff On Runway button (). The Takeoff On Runway dialog box opens.
3. In the Airport box, enter the ICAO code of the aerodrome. You can type the code, select a managed aerodrome in the Objects List, or select from the list of aerodromes at the bottom of the window. In the last case, any managed aerodromes are listed at the top of the list, and then all other aerodromes in the terrain are listed below those.
4. Select the specific runway from which the entity should take off. The runways listed are those associated with the selected aerodrome.
5. Optionally, select the Teleport to Runway Start check box to have the entity move immediately to the start of the runway. This saves some time by quickly positioning the entity for takeoff.
6. Click OK.

➤ **To assign a Takeoff On Runway task for the closest runway:**

1. Select the entity.
2. Choose **Task > Movement > Takeoff On Runway**, or click the Takeoff On Runway button (). The Takeoff On Runway parameters display.
3. Select the Use Closest Runway check box.
4. Optionally, select the Teleport to Runway Start check box.
5. Click OK.

You can explicitly task a fixed-wing entity to take off. For details about innate take-off, see "24.1.5 How Fixed-Wing Entities Take Off" on page 511.

30.51. Takeoff (Vertical)

Aircraft configured with the Fixed Wing VTOL movement system can perform a vertical takeoff task.

➤ **To assign a Takeoff (Vertical) task:**

1. Select the entity.
2. Choose **Task > Movement > Takeoff (Vertical)**. The Takeoff (Vertical) parameter displays.
3. Enter the height to which the entity should move.
4. Click OK.

30.52. Transition Into Formation

You can order a ground unit to transition to a formation and heading. They travel a short distance in the heading specified to assume the formation, but remain in generally the same location. If you want the unit to move to another location and assume a formation, see "30.26. Move Into Formation" on page 603.

Transitioning to a formation is not the same as setting formation. Setting formation changes the formation instantly, as described in "40.33. Formation" on page 879.

➤ **To assign a Transition Into Formation task:**

1. Select the unit.
2. Choose **Task > Movement > Transition Into Formation**. The Transition Into Formation parameters display.
3. Select the desired formation from the list.
4. Specify the heading the unit should face after assuming the specified formation.
5. Click OK.

30.53. Turn to Heading

The Turn to Heading task provides a realistic way for ground vehicles to turn to a new heading. Rather than turning instantly, as happens with the Heading set data request, a vehicle pivots or makes a K-turn to the new heading. The turning behavior for an entity is controlled by its **can-pivot** parameter in the object parameter database.

➤ **To assign a Turn to Heading task:**

1. Select the entity.
2. Choose **Task > Movement > Turn to Heading**. The Turn to Heading parameters display.
3. Type a heading.
4. In the Relative To list, select one of the options:
   - True North. Turn to the specified absolute heading (true north).
   - Host Object (if embarked). If embarked, turn to the specified heading relative to the host simulation object.
   - Current Heading. Turn the specified number of degrees relative to the current heading.
   - Magnetic North. Turn to the specified heading relative to magnetic north.
5. Click OK.

30.54. Zig-Zag to Heading

The Zig Zag to Heading task causes a surface watercraft entity to move towards a heading using a zig-zag pattern.

➤ **To assign a Zig-Zag to Heading task:**

1. Select the entity.
2. Choose **Task > Movement > Zig-Zag to Heading**. The Zig-Zag to Heading parameters display.
3. Specify the heading.
4. Specify the amount of time (in minutes) to spend on each leg of the zig-zag pattern.
5. Specify the angle of direction for each leg.



31. Engagement Tasks for Entity-Level Scenarios

This chapter describes tasks in the engagement, offense, and defense categories that apply to entity-level scenarios.

31.1. Arm Mine at Depth 624
31.2. Attack Aircraft 625
31.3. Attack Aircraft with Guns 625
31.4. Cease Fire 625
31.5. Drop Naval Depth Charge 625
31.6. Drop Naval Depth Charge at Location 626
31.7. Drop Naval Mine 626
31.8. Drop Naval Mines Along Route 628
31.9. Execute Gun Attack 628
31.10. Execute Surface Attack 630
31.11. Explode Charge at Depth 631
31.12. Fire At 632
31.13. Fire Ballistic Gun 632
31.14. Fire Cruise Missile 633
31.15. Fire for Effect 634
    31.15.1 Firing Naval Guns 635
31.16. Fire Javelin at Target 635
31.17. Fire Laser Guided Missile 636
31.18. Fire Missile Launcher 636
31.19. Fire Team Suppress Target 637
31.20. Lase and Fire Guided Missile 637
31.21. Lase Target 637
31.22. Launch Anti-Submarine Missile (Vertical) 638
31.23. Launch Ballistic Missile at Location 639
31.24. Launch Ballistic Missile at Target Object 639
31.25. Launch Countermeasures 640

31.26. Launch Smoke 640
31.27. Launch Torpedo Tasks 641
31.27.1 Anti-ship (Fixed Depth) 641
31.27.2 Anti-Submarine 641
31.28. Loiter Munition Tasks 642
31.28.1 Launch Loiter Munition 642
31.28.2 Loitering Attack 642
31.29. Place IED 643
31.30. Provide Indirect Fire 644
31.31. Provide Suppressive Fire 645
31.32. Release Bomb Tasks 646
31.33. Release Unguided Bomb 647
31.34. Throw Grenade Tasks 648
31.34.1 Throw Grenade at Location 648
31.34.2 Throw Grenade at Target 648

31.1. Arm Mine at Depth

If a naval mine has been deployed, Arm Mine at Depth allows you change the mine's parameters.

Arm Mine at Depth is a system scripted task. It is disabled by default.

➤ **To assign an Arm Mine at Depth task:**

1. Select the mine.
2. Choose **Task > Arm Mine at Depth**. The Arm Mine at Depth parameters display.
3. Specify the depth at which to arm the mine.
4. Specify the radius within which target entities will detonate the mine.
5. Optionally, specify a time when the mine will be armed, in days and hours:minutes:seconds.
6. In the Trigger Entity Types list, add the types of entities that can trigger the mine:
    a. Click the Add button (). The Entity Type dialog box opens.
    b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration.
    c. Click OK.
7. To specify that only hostile entities will set off the mine, select Hostile Only. If this setting is cleared, any entity will set off the mine.
8. Click OK.

31.2. Attack Aircraft

Use this task to cause one aircraft to attack another one. The aircraft entity assigned this task will fly toward and fire on the target. The aircraft must be configured with a weapon that is capable of attacking the target aircraft.

➤ **To assign an Attack Aircraft task:**

1. Select an aircraft entity that has a weapon system.
2. Choose **Task > Engagement > Attack Aircraft**. The Attack Aircraft parameters display.
3. Select the target entity.
4. Click OK.

31.3. Attack Aircraft with Guns

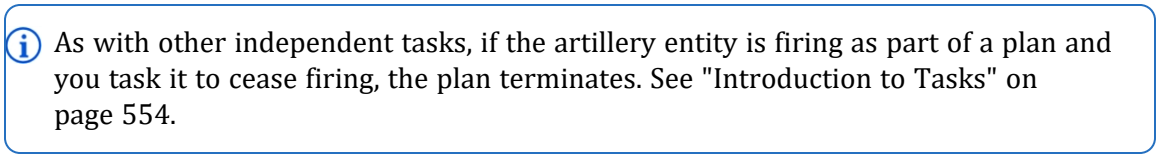
The Attack Aircraft with Guns task causes aircraft that have mounted guns, such as the A-10 Thunderbolt, to target a specified aircraft with its guns rather than air-to-air missiles.

➤ **To assign an Attack Aircraft with Guns task:**

1. Select an aircraft entity that has a gun.
2. Choose **Task > Engagement > Attack Aircraft with Guns**. The Attack Aircraft with Guns parameters display.
3. Select the target entity.
4. Click OK.

31.4. Cease Fire

The Cease Fire task passes a cease fire order to artillery systems.



As with other independent tasks, if the artillery entity is firing as part of a plan and you task it to cease firing, the plan terminates. See "Introduction to Tasks" on page 554.

➤ **To assign the Cease Fire task:**

1. Select the entity that you want to stop firing.
2. Choose **Task > Engagement > Cease Fire**.

31.5. Drop Naval Depth Charge

The Drop Naval Depth Charge task causes a fixed-wing, rotary wing, or surface entity to drop a depth charge at its current location. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system.

➤ **To assign a Drop Naval Depth Charge task:**

1. Select the entity.
2. Choose **Task > Engagement > Drop Naval Depth Charge**. The Drop Naval Depth Charge parameters display.
3. Select the type of depth charge to drop.
4. Specify the depth at which you want the depth charge to explode.
5. Click OK.

31.6. Drop Naval Depth Charge at Location

The Drop Naval Depth Charge at Location task causes a fixed-wing, rotary wing, or surface entity to move to the specified location and drop a depth charge. To execute this task, an entity must have a Naval Depth Charge Deployment weapon system.

ⓘ In this task, the location is the location of the entity when it drops the depth charge. Since the entity is flying, the location must include a reasonable altitude.

➤ **To assign a Drop Naval Depth Charge at Location task:**

1. Select the entity.
2. Choose **Task > Engagement > Drop Naval Depth Charge at Location**. The Drop Naval Depth Charge At Location parameters display.
3. In the Naval Depth Charge list, select the type of depth charge to drop.
4. Specify the depth at which you want the depth charge to explode.
5. Specify the location where you want to drop the depth charge. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.

   By default, this task uses the altitude of the entity at the time the task is assigned as the altitude for the drop location. (Use Current Altitude check box.) If you specify an altitude and the check box is selected, VR-Forces ignores the altitude you entered and uses the current entity altitude. If you want to specify the altitude, clear the check box and make sure that you specify the altitude in the Altitude group box. If you click on the terrain to specify the X, Y location, the altitude is set to 0, so be sure to enter the altitude you want after clicking on the terrain.
6. Click OK.

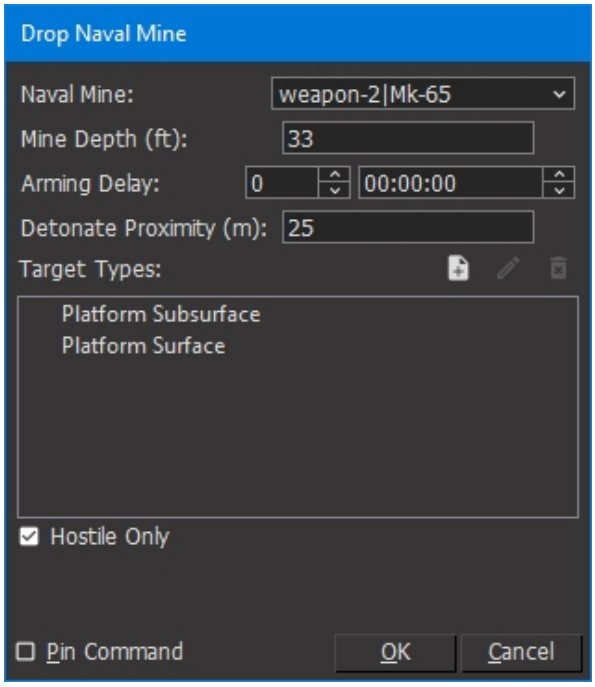
31.7. Drop Naval Mine

The Drop Naval Mine task allows entities with a Naval Mine Deployment system (fixed-wing, surface, and subsurface) to drop naval (underwater) mines at their current location.

➤ **To assign a Drop Naval Mine task:**

1. Select the entity.
2. Choose **Task > Engagement > Drop Naval Mine**. The Drop Naval Mine parameters display (Figure 237).

Figure 237. Drop Naval Mine dialog box



3. In the Naval Mine list, select the type of mine to drop.
4. In the Mine Depth field, set the depth to drop the mine. Positive numbers are below sea level.
5. In the Arming Delay fields, set the time to elapse before the mine becomes armed in days and hours:minutes:seconds.
6. In the Detonate Proximity field, set the distance from the mine within which a target entity will detonate the mine.
7. In the Target Types list, add the types of entities that can trigger the mine:
   a. Click the Add button ( ). The Entity Type dialog box opens.
   b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration. By default, surface and subsurface watercraft are targets.
   c. Click OK.
8. To specify that only hostile entities will set off the mine, select Hostile Only. If this setting is cleared, any entity will set off the mine.
9. Click OK.

31.8. Drop Naval Mines Along Route

The Drop Naval Mines Along Route task allows entities with a Naval Mine Deployment system (fixed-wing, surface, and subsurface) to drop naval (underwater) mines while they travel along a route.

➤ **To assign a Drop Naval Mines Along Route task:**

1. Select the entity.
2. Choose **Task > Engagement > Drop Naval Mines Along Route**. The Drop Naval Mines Along Route parameters display.
3. Select the route to follow.
4. In the Mine Spacing field, specify the distance between each mine.
5. In the Naval Mine list, select the type of mine to drop.
6. In the Mine Depth field, set the depth to drop the mine. Positive numbers are below sea level.
7. In the Arming Delay fields, set the time to elapse before the mine becomes armed in days and hours:minutes:seconds.
8. In the Detonate Proximity field, set the distance from the mine within which a target entity will detonate the mine.
9. In the Trigger Entity Types list, add the types of entities that can trigger the mine:
    a. Click the Add button (). The Entity Type dialog box opens.
    b. Build an entity type by selecting options from the lists for each enumeration component or type an enumeration. By default, surface and subsurface watercraft are targets.
    c. Click OK.
10. To specify that only hostile entities will set off the mines, select Hostile Only. If this setting is cleared, any entity will set off the mine.
11. Click OK.

31.9. Execute Gun Attack

Fixed-wing entities that have the appropriate armament can strafe ground targets. The Execute Gun Attack task takes these parameters:

- Initial Point. The point at which the aircraft reaches strafing altitude and lines up to strafe the target point. VR-Forces enforces a minimum distance from the initial point to the target. It varies based on the plane's dive angle, strafe range, entry speed, and other parameters. Two kilometers is a good minimum for most entities. If the initial point is too close to the target, VR-Forces sends an error message to the entity console.

- Target. The entity or object to target. If you specify none, the entity strafes the target location. If you specify an object type, the entity looks for that object type. If it finds the target type, it strafes it. If it does not find the target type, it either strafes the target point (default behavior) or aborts the strafing run, depending on the state of the Abort Without Target check box.
- Egress Heading and Egress Maneuver. The way the aircraft should exit the strafing run and the heading at which it exits.
- Specify Time on Target.

You can also initiate a strafing run from the Execute Surface Attack task. Table 31 shows how arguments to the Execute Surface Attack task map to the Execute Gun Attack task.

Table 31. Surface to Gun Attack argument mapping

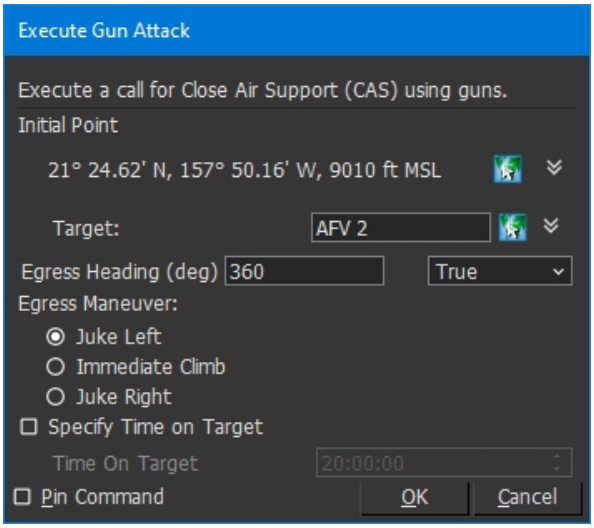
Execute Surface Attack Argument	Execute Gun Attack Argument
Initial Point	Initial Point.
Target: an entity	Target Type: Target the indicated entity.
Offset: none	Egress Maneuver: Immediate Climb.
Offset: left	Egress Maneuver: Juke Left.
Offset: right	Egress Maneuver: Juke Right.
	Abort without target.
Egress Heading	Final Egress Heading.

The effectiveness of a strafing run at killing entities around the target point depends in part on the value of the **dispersal-angle** parameter specified for the munition. You can edit this parameter in *./data/simulationModelSets/base/detonationParams.mtl*. The parameter is described in the comments at the top of the file.

➤ **To assign an Execute Gun Attack task:**

1. Select the entity that will make the strafing run.
2. Choose **Task > Engagement > Execute Gun Attack**. The Execute Gun Attack parameters display (Figure 238).

Figure 238. Execute Ground Attack dialog box



3. Click the terrain to specify the initial point or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Click the Select Object button ( ) and then specify the target entity.
5. Select an egress maneuver.
6. Optionally, specify an egress heading.
7. Optionally, select Specify Time on Target and specify the time at which the entity should fire upon the target. This controls the entity's speed as it approaches the drop site.
8. Click OK.

31.10. Execute Surface Attack

The Execute Surface Attack task assigns a fixed-wing fighter or bomber to release bombs on a ground target using coordinates, a laser designation code, or a specific target. If you specify multiple targets, all bombs are dropped simultaneously. The aircraft does not fly over each target. If you specify an initial point, that point displays as part of the task visualization when you select the entity.

Due to the Time on Target option, this task is not intended for use with Move Along Route with Actions. It is intended to be a single use for dropping bombs on targets.

➤ **To assign an Execute Surface Attack task:**

1. Select the entity you want to execute the surface attack.
2. Choose **Task > Engagement > Execute Surface Attack**, or click the Execute Surface Attack button ( ). The Execute Surface Attack parameters display.

3. Optionally, select Initial Point and then enter the coordinates. (The coordinate system is specified in Display Units Settings.)
4. Click the Add button (+) to add a target to the list. The Target Configuration dialog box opens.
5. Choose the Target Type.
   - MGRS Coordinates or Lat/Lon Coordinates. Enter the Location coordinates.
   - Laser Designation Code. Enter a code.
   - Target. Use the target box to select the desired target.
6. Select the Weapon (Bomb) from the list.
7. Click OK. You return to the Execute Surface Attack, where the target is shown in the list.
8. You can add multiple targets by repeating steps 4–7.
9. Optionally, specify the Time on Target to indicate the time at which the entity should drop the bombs on the targets. This controls the entity's speed as it approaches the drop site.
10. Optionally, specify the Egress Heading.
11. Optionally, enter a Release Speed and Release Altitude. The speed is only the speed at the time of the drop, not the speed for the duration of the task. During the rest of the task, it uses the standard or ordered speed.
12. Click OK.

31.11. Explode Charge at Depth

The Explode Charge at Depth task explodes a depth charge at the specified depth. You can use this task to explode a depth charge that has been deployed, but has not yet exploded.

Explode Charge at Depth is a system scripted task. It is disabled by default.

➤ **To assign an Explode Charge at Depth task:**

1. Select the depth charge.
2. Choose **Task > Explode Charge at Depth**. The Explode Charge at Depth parameters display.
3. Specify the depth.
4. Click OK.

31.12. Fire At

The Fire At task commands an entity to fire at a specific target, either a location or simulation object. You can specify which weapon system to use if the entity has more than one.

When you assign this task, the entity ignores its rules of engagement and force types. Entities will fire at their own force if told to do so. The shooter must have an appropriate weapon system and ammunition for the target, and also must be in a position to fire at the target. If the entity cannot fire on the target, it abandons the task.

Note that, if you do not specify a specific weapon, the weapon shown in the disabled Preferred Weapon list may not be the weapon that the entity uses to fire at the target.

➤ **To assign a Fire At task:**

1. Select the entity you want to fire.
2. Choose **Task > Engagement > Fire At**. The Fire At parameters display.
3. Click Select Location or Select Object.
4. Select the target:
   - Click the location to fire at or enter the coordinates.
   - Select the simulation object.
5. Enter the number of rounds you want the entity to fire.
6. Optionally, select the Stop Firing When Target is Destroyed check box. This causes the entity to stop firing when the target is destroyed or when the maximum number of rounds has been fired, whichever comes first.
7. Optionally, select the Prefer Specific Weapon check box and then choose one from the list of the entity's available weapons.
8. Click OK.

31.13. Fire Ballistic Gun

The Fire Ballistic Gun task commands an entity to fire at a specific target, either a location or simulation object. You can specify which weapon system to use and the maximum number of rounds to fire. The entity must be equipped with a ballistic weapon system.

When you assign this task, the entity ignores its rules of engagement and force types. Entities will fire at their own force if told to do so.

➤ **To assign a Fire Ballistic Gun task:**

1. Select the entity you want to fire.
2. Choose **Task > Engagement > Fire Ballistic Gun**. The Fire Ballistic Gun parameters display.
3. Click Select Location or Select Object.

4. Select the target:
   - Click a location on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
   - Select the simulation object.
5. If the entity has more than one weapon system equipped, select the weapon you want the entity to fire.
6. Enter the maximum number of rounds the entity should fire.
7. Click OK.

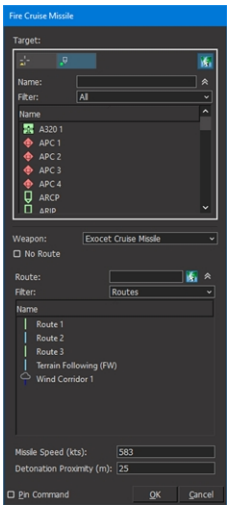
31.14. Fire Cruise Missile

Entities with a cruise missile launcher system can fire cruise missiles. A cruise missile can fly directly to a target or follow a route.

➤ **To assign a Fire Cruise Missile task:**

1. Select an entity that has cruise missiles.
2. Choose **Task > Engagement > Fire Cruise Missile**. The Fire Cruise Missile parameters display (Figure 239).

Figure 239. Fire Cruise Missile dialog box



3. Click Select Location or Select Object.
4. Select the target:
   - Click the location to fire at or enter the coordinates.
   - Select the simulation object.
5. If the entity is equipped with more than one weapon system, select the weapon that you want the entity to fire.
6. Select a route for the missile to follow or select the No Route check box. If you do not select a route, the missile flies directly to the target.
7. Specify the Missile Speed.

8. Specify the Detonation Proximity for the warhead. It will detonate within this range of the selected target.
9. Click OK.

31.15. Fire for Effect

You can task artillery entities and surface vessels with naval guns to fire for effect (FFE).

When the entity executes the task, you can select it to see a countdown to when it will fire its weapon. After it fires, there is a countdown to the splash at the specified location, entity, or point.

➤ **To assign a Fire for Effect task:**

1. Select the entity to which you want to assign the task.
2. Choose **Tank > Engagement > Fire for Effect**. The Fire for Effect parameters display.
3. Click Select Location or Select Object.
4. Specify the target:
   - When firing on a location, click the location on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
   - When firing on a unit, entity, or point, select it from the list in the parameters, the Objects List, or on the terrain. You can filter the list as desired.
5. If the entity is equipped with more than one weapon system, select the weapon that you want the entity to fire.
6. Select the munition.
7. Indicate how you want the entity to fire:
   - Specify the number of rounds.
   - Select Continuous to fire continuously until the target is destroyed or the entity has no more ammunition.
8. Optionally, specify the height above the terrain at which rounds should detonate.
9. Click OK.

- The Fire-for-Effect task is meaningful only when assigned to artillery entities or surface vessels that have naval guns.
- When you choose to fire on an entity and the target entity is moving, VR-Forces dead reckons its position and the artillery fires at its expected location, not the location it was at the start of the task.
- Artillery entities have maximum and minimum ranges. To see these ranges, enable range rings for the entity. (For details, see "50.8. Displaying Threat and Sensor Range Rings" on page 1065.) The sim engine console window prints messages when you select targets that are outside of the entity's range.
- The default preferred trajectory height for artillery and naval guns is 500 meters. If this height is not attainable given other parameters involved in calculating the trajectory, VR-Forces calculates a more appropriate height. You can specify a different preferred trajectory height with the Preferred Trajectory Height set data request. For details, see "40.60. Preferred Trajectory Height" on page 893.

31.15.1 Firing Naval Guns

Ordinarily, naval guns fire their munitions in a parabola, similar to artillery. However, they can also fire directly at a target if these conditions are true:

- The target's range is greater than the weapon's minimum range, but less than the first value in its range list.
- The **capable-of-direct-alignment** parameter for the weapon system's naval-gun-controller is set to True.
- The firing entity has line-of-sight to the target.

31.16. Fire Javelin at Target

This task causes a human entity equipped with a Javelin to fire it at a target. You can choose either Top Attack or Direct Attack, depending how you want the missile to fly at the target. The entity takes the correct position, sitting cross-legged, to fire the weapon, and then it stands again. Due to how the Javelin functions, the entity must have line of sight to the target or else it will not fire the weapon.

➤ **To assign the Fire Javelin at Target task:**

1. Select the entity.
2. Choose **Task > Engagement > Fire Javelin at Target**. The Fire Javelin at Target parameters display.
3. In the Target box, enter the simulation object.
4. From the Attack Mode list, select the desired trajectory.
5. Click OK.

31.17. Fire Laser Guided Missile

This task causes an entity equipped with laser-guided missiles to launch a missile without lasing the target object or location. It is designed for use when a separate entity will lase the target, however it can fire the missile without a laser designator.

By default, the entity firing the missile waits to detect a designator before firing. You must specify the laser code that it will target, as described in "40.46. Laser Code" on page 885.

You can specify that the entity fire the missile without a laser designator, in which case the entity immediately fires the missile.

➤ **To assign the Fire Laser Guided Missile task:**

1. Select the entity.
2. Choose **Task > Engagement > Fire Laser Guided Missile**. The Fire Laser Guided Missile task parameters display.
3. From the Weapon Selection list, choose the missile that you want the entity to fire.
4. Optionally, select the Fire Without Designator check box.
5. Click OK.

31.18. Fire Missile Launcher

The Fire Missile Launcher task commands an entity to launch a missile at a specific target, either a location or simulation object. You can specify which missile launcher system to use and the maximum number of rounds to fire. The entity must be equipped with a missile launcher system.

When you assign this task, the entity ignores its rules of engagement and force types. Entities will fire at their own force if told to do so. The shooter must have an appropriate weapon system and ammunition for the target.

ⓘ Cruise missiles are special, in that they can follow a specific route. If that is the type of missile you want to fire, see "31.14. Fire Cruise Missile" on page 633.

➤ **To assign a Fire Missile Launcher task:**

1. Select the entity you want to fire.
2. Choose **Task > Engagement > Fire Missile Launcher**. The Fire Missile Launcher parameters display.
3. Click Select Location or Select Object.
4. Select the target:
   - Click the location to fire at or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
   - Select the simulation object.

5. If the entity is equipped with more than one missile system, select the launcher you want the entity to fire.
6. Enter the maximum number of rounds the entity should fire.
7. Click OK.

31.19. Fire Team Suppress Target

The Fire Team Suppress Target task tells a fire team to fire at a simulation object such as a point, line, or area, one meter above the terrain. If one of the entities in the fire team must stand or move to the side to see the target, it does so. For more information, see "26.1. Suppression" on page 520.

➤ **To assign a Fire Team Suppress Target task:**

1. Select a fire team.
2. Choose **Task > Engagement > Fire Team Suppress Target**. The Fire Team Suppress Target parameters display.
3. Select the simulation object to fire at.
4. In the Rapid Fire Duration box, enter the number of seconds for the rapid fire.
5. In the Total Fire Duration box, enter the number of minutes the entity should provide suppressive fire.
6. Click OK.

31.20. Lase and Fire Guided Missile

This causes an entity equipped with laser-guided missiles to lase a target location or simulation object and then fire a missile at it.

➤ **To assign the Lase and Fire Guided Missile task:**

1. Select the entity.
2. Choose **Task > Engagement > Lase and Fire Guided Missile**. The Lase and Fire Guided Missile task parameters display.
3. Click Select Location or Select Object.
4. Select the target:
   - Click the location to lase and target or enter the coordinates.
   - Select the simulation object.
5. Click OK.

31.21. Lase Target

The Lase Target task simulates aiming a laser beam at a target or location to guide a laser guided weapon. Use this task when you want to lase a specific simulation object or spot on the terrain. An entity can lase a target while executing a movement task.

To lase a target, an entity must have a Laser Designator system. For more information about lasing targets, see:

- "9.4.2 Lasing Targets" on page 304.
- "40.46. Laser Code" on page 885.
- "40.83. Synchronize Laser Code" on page 903.

- The Lase Target task takes line of sight into account when looking at a target object or location. Obstacles can obstruct the laser beam. If there are obstacles between the lasing entity and the specified simulation object, the task is unsuccessful. If there are obstacles between the lasing entity and the specified location, the laser designator spot is at the point blocking the line of sight.
- When an entity lases one side an entity, the direction of the laser is considered and weapons can only target the designated spot if they have line of sight to it. If the weapon does not have visibility to that side of the entity, the weapon cannot target it.
- Laser designator systems have a maximum range. If the entity aiming the laser designator is too far away from the location it is trying to lase, the task is unsuccessful.

➤ **To assign a Lase Target task:**

1. Select the entity that will aim the laser.
2. Choose **Task > Engagement > Lase Target**. The Lase Target parameters display.
3. Click Select Location or Select Object.
4. Specify the target:
   - Click the location on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
   - Select the simulation object from the list in the parameters, the Objects List, or on the terrain. You can filter the list as desired.
5. Click OK.

31.22. Launch Anti-Submarine Missile (Vertical)

The Launch Anti-Submarine Missile (Vertical) task commands a surface entity to launch a missile that flies to the vicinity of the target and then drops a homing torpedo. To execute this task, an entity needs an Anti-Submarine Missile (Vertically Launched) weapon system.

➤ **To assign a Launch Anti-Submarine Missile (Vertical) task:**

1. Select the surface entity.
2. Choose **Task > Engagement > Launch Anti-Submarine Missile (Vertical)**. The Launch Anti-Submarine Missile (Vertical) parameters display.
3. Select the target entity.

4. Specify the location at which to drop the homing torpedo.
5. Click OK.

31.23. Launch Ballistic Missile at Location

The Launch Ballistic Missile at Location task lets an entity that has a strategic missile weapon system launch a missile at a location.

> The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

➤ **To launch a ballistic missile at a location:**

1. Select an entity with a strategic missile weapon system.
2. Choose **Task > Engagement > Launch Ballistic Missile at Location**. The Launch Ballistic Missile at Location parameters display.
3. Click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. In the Missile Type list, select the missile to fire.
5. Click OK.

31.24. Launch Ballistic Missile at Target Object

The Launch Ballistic Missile at Target Object task lets an entity that has a strategic missile weapon system launch a missile at a point object, entity, or unit. The destination of the missile is the target's location when the missile is fired. The missile does not track moving targets.

> The Ballistic Missile tasks do not use physics-based modeling. They calculate a simple parabola for the trajectory and move the missile along the trajectory ignoring atmospherics, thrust stages, cargo weight, gravity, and the rotation of the earth.

➤ **To launch a ballistic missile at a target object:**

1. Select an entity with a strategic missile weapon system.
2. Choose **Task > Engagement > Launch Ballistic Missile at Target Object**. The Launch Ballistic Missile at Target Object parameters display.
3. Optionally, filter the list of objects.
4. Select the object to fire at.
5. In the Missile Type list, select the missile to fire.
6. Specify the altitude where the missile should detonate.
7. Click OK.

31.25. Launch Countermeasures

Fixed-wing and rotary-wing entities can launch countermeasures, such as chaff and flares. Launch Countermeasures is not an exclusive task. It does not interrupt an entity's current task. (For more information, see "9.9. Launching Countermeasures (Chaff and Flare)" on page 311.)

The Evade Missile reactive task automatically launches chaff and flare when a fixed-wing entity evades a missile.

➤ **To assign a Launch Countermeasures task:**

1. Select the entity that you want to launch countermeasures.
2. Choose **Task > Engagement > Launch Countermeasures**. The Launch Countermeasures parameters display.
3. In the Countermeasure Types list, select the type of countermeasures you want to launch.
4. In the Number To Fire box, specify the number of countermeasures to launch.
5. In the Time Between Fires box, specify the amount of time, in seconds, between each countermeasure launch.
6. Click OK.

31.26. Launch Smoke

The Launch Smoke task causes an entity to create a smoke cloud. The cloud is created as an environmental object. It grows over time.

An entity can launch a smoke cloud towards a threat or in a specific direction. Smoke clouds affect sensors. They do not affect entity movement.

- Tactical smoke is delivered by the M250 smoke grenade launcher. An entity must have this system to execute this task.
- Wind and precipitation dissipate tactical smoke. You can adjust those dissipation factors by editing the Tactical Smoke simulation object in the Simulation Object Editor.

➤ **To assign a Launch Smoke task:**

1. Select the entity that you want to launch smoke.
2. Choose **Task > Engagement > Launch Smoke**. The Launch Smoke parameters display.
3. Select At Nearest Threat or Toward Heading.
4. If you selected Toward Heading, specify a compass heading and indicate whether it is relative to true or magnetic north.
5. Click OK.

31.27. Launch Torpedo Tasks

Subsurface entities can launch two types of anti-ship torpedoes: one moves at a fixed depth, and the other homes in on the center of the target. In both cases, the torpedo travels at an initial bearing for a fixed distance before it homes in on the target.

> Torpedoes rise to target depth at a fixed rate. If a submarine is quite close to a target when it fires, the torpedo might not be able to rise to target depth in the distance from the submarine to the target. If this happens, it circles until it reaches target depth.

31.27.1 Anti-ship (Fixed Depth)

➤ **To assign a Launch Torpedo – Anti-ship (Fixed Depth) task:**

1. Select a subsurface entity.
2. Choose **Task > Engagement > Launch Torpedo > Anti-ship (Fixed Depth)**. The Anti-ship (Fixed Depth) parameters display.
3. Optionally, filter the entity list.
4. Select the target entity.
5. In the Initial Bearing box, specify the bearing the torpedo should travel at until it begins homing in on the target.
6. In the Homing Start Delay box, enter the amount of time that the torpedo should travel on its initial bearing.
7. In the Cruise Depth box, specify the depth the torpedo should travel at. It will remain at this depth.
8. In the Proximity Trigger Distance box, set the distance from a target at which the torpedo will detonate.
9. Click OK.

31.27.2 Anti-Submarine

The anti-submarine torpedo homes in on the center of the target.

➤ **To assign a Launch Torpedo – Anti-submarine task:**

1. Select a subsurface entity.
2. Choose **Task > Engagement > Launch Torpedo > Anti-submarine**. The Anti-Submarine parameters display.
3. Optionally, filter the entity list.
4. Select the target entity.
5. In the Initial Bearing box, specify the bearing the torpedo should travel at until it begins homing in on the target.

6. In the Homing Start Delay box, enter the amount of time that the torpedo should travel on its initial bearing.
7. In the Proximity Trigger Distance box, set the distance from the target at which the torpedo will detonate.
8. Click OK.

31.28. Loiter Munition Tasks

31.28.1 Launch Loiter Munition

Entities configured with weapon systems for loiter munitions, such as the Switchblade 300 and 600 Launchers, can launch the munitions that will then orbit a location or strike a simulation object. When you use delayed targeting, you must then use the Target set data request after the munition is launched to specify which simulation object the munition should strike. See "40.86. Target" on page 905.

➤ **To assign the Launch Loiter Munition task using Delayed Targeting:**

1. Select the launcher entity.
2. Choose **Task > Engagement > Launch Loiter Munition**. The Launch Loiter Munition parameters display.
3. Select Delayed Targeting and then specify the location that the munition should orbit.
4. Click OK. The munition is launched and then moves to the correct altitude and position to conduct its orbit task.
5. When you want the loiter munition to strike, select the munition.
6. Choose **Set > Engagement > Target** and then specify the simulation object. After you click OK, the loiter munition flies toward and crashes into the target.

➤ **To Assign the Loitering Attack task using Immediate Targeting:**

1. Select the launcher entity.
2. Choose **Task > Engagement > Launch Loiter Munition**. The Launch Loiter Munition parameters display.
3. Select Immediate Targeting and then specify the simulation object that the munition should strike.
4. Click OK. The munition is launched, then flies toward and crashes into the target.

31.28.2 Loitering Attack

Loiter munitions and drones equipped with munitions can perform the Loitering Attack task.

You can task a loiter munition to orbit a location or strike a simulation object. When you use delayed targeting, you must then use the Target set data request to tell the munition which simulation object to strike when you want it to attack. See "40.86. Target" on page 905.

Drone munitions can only perform immediate targeting.

Loiter Munitions

Loiter munitions include the Switchblade 300 or 600.

➤ **To assign the Loitering Attack task using Delayed Targeting:**

1. Select the loiter munition entity.
2. Choose **Task > Engagement > Loitering Attack**. The Loitering Attack parameters display.
3. Select Delayed Targeting and then specify the location that the munition should orbit.
4. Click OK. The munition moves to the correct altitude and position to conduct its orbit task.
5. When you want the loiter munition to strike, select the munition if it is not still selected.
6. Choose **Set > Engagement > Target** and then specify the simulation object. After you click OK, the loiter munition flies toward and crashes into the target.

➤ **To Assign the Loitering Attack task using Immediate Targeting:**

1. Select the loiter munition entity.
2. Choose **Task > Engagement > Loitering Attack**. The Loitering Attack parameters display.
3. Select Immediate Targeting and then specify the simulation object that the munition should strike.
4. Click OK. The munition flies toward and crashes into the target.

Drones with Munitions

Drones that can perform the Loitering Attack task include the DJI Mavic 3 Small Munition and DJI Mavic 3 Small Munition Fiber Optic.

➤ **To Assign the Loitering Attack task:**

1. Select the drone entity.
2. Choose **Task > Engagement > Loitering Attack**. The Loitering Attack parameters display.
3. Specify the simulation object that the munition should strike.
4. Click OK. The drone flies toward and crashes into the target.

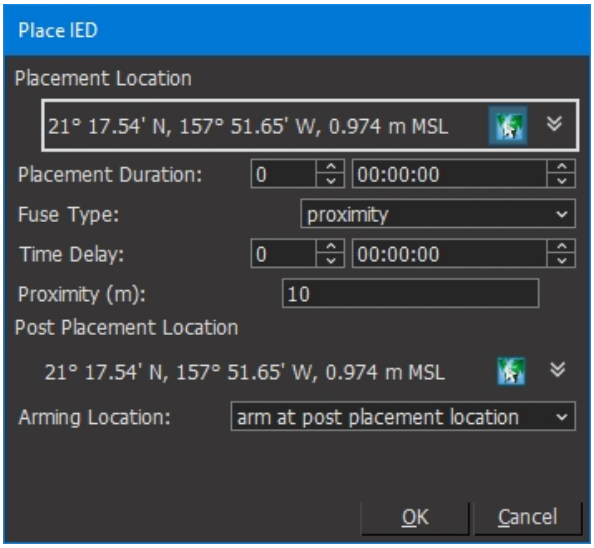
31.29. Place IED

The Place IED task lets a human entity place an improvised explosive device (IED) at a particular location and move to a post-placement location. This task also sets the parameters for how the IED is armed.

➤ **To assign a Place IED task:**

1. Select the entity.
2. Choose **Task > Engagement > Place IED**. The Place IED parameters display (Figure 240).

Figure 240. Place IED dialog box



3. Click on the terrain where the entity should place the IED, or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. In the Placement Duration box, enter the amount of time it takes to place the IED.
5. Select a fuse type from the Fuse Type list.
6. For a timed delay fuse, specify the time to delay in the Time Delay box.
7. For a proximity fuse, specify the distance from the IED within which another entity will trigger the IED.
8. For the Post Placement Location, specify the location where the entity should move to after it places the IED, or click on the terrain to specify the location.
9. From the Arming Location list, select Arm at Placement Location to arm the IED as soon as it is placed, or Arm at Post Placement Location to arm it when the entity reaches the post-placement location.
10. Click OK.

31.30. Provide Indirect Fire

The Provide Indirect Fire task allows you to task artillery and surface watercraft entities that have naval guns to provide indirect fire. It was designed for an artillery officer who would fill in the information based on communication of a Call for Fire 6 line.

> The task was designed for specific use cases. It does not include some information such as a Warning Order, which can be understood in the mind of the person filling in the task's options.

There are a number of targeting methods available to choose:

- Coordinates. Specify either MGRS or Lat/Lon (DMS)coordinates, either by entering the information or clicking on the terrain.
- Polar. Select a Forward Observer entity and enter the location of the target in direction and distance from the observer.
- Adjust. Select a forward observer and enter the adjustment and direction and distance from the previously targeted point.
- Known Point. Use a known target point Specify the Forward Observer, Known Point, and Adjustment from the known point location based on the forward observer location. This require s a known target point, which is created by a previous run of the Provide Indirect Fire task with the Generate Known Target Point check box selected.
- Target. Select a simulation object.

➤ **To assign a Provide Indirect Fire task:**

1. Select the entity that will provide indirect fire.
2. Choose **Task > Engagement > Provide Indirect Fire**, or click the Indirect Fire button (). The Provide Indirect Fire parameters display.
3. Click the button for the method that you want to use to enter the target location.
4. Enter the parameters for the targeting method you selected.
5. Specify the munition.
6. Enter the number of Rounds to fire, or select the Continuous check box to fire until the entity runs out of ammunition for the selected weapon.
7. Choose the Fuse type. If you use Other, enter the Height Above Terrain.
8. Optionally, enter a Fire Mission Index. This is a unique identifier for the network PDU. It does not affect the VR-Forces task.
9. Optionally, select the Generate Known Target Point check box. This creates a target point with a 2-letter and 4-number designation, which is used by the Adjust target method.
10. Click OK.

31.31. Provide Suppressive Fire

The Provide Suppressive Fire task causes an entity to fire on a location, or on a simulation object such as a point, line, or area. It fires 1 meter above the specified target. When targeting on a simulation object, if it is in cover, the entity moves 1 meter to the left or right or stands up. For more information, see "26.1. Suppression" on page 520.

> The target of suppressive fire must be at least 25 meters away.

➤ **To assign a Provide Suppressive Fire at Object task:**

1. Select the entity that will deliver suppressive fire.
2. Choose **Task > Engagement > Provide Suppressive Fire**. The Provide Suppressive Fire parameters display.
3. Click Select Location or Select Object.
4. Specify the target to suppress:
   - When firing on a location, click the location on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
   - When firing on a simulation object, select it from the list in the parameters, the Objects List, or on the terrain. You can filter the list as desired.
5. In the Rapid Fire Duration box, enter the number of seconds for the rapid fire.
6. In the Total Fire Duration box, enter the number of seconds the entity should provide suppressive fire.
7. From the Gun to Use list, select the weapon to prefer.
8. Enter the maximum number of rounds for the entity to fire.
9. Click OK.

31.32. Release Bomb Tasks

Fixed-wing entities can release bombs targeted to a laser spot, a location, or an entity. The Release Bomb task simply causes the entity to immediately drop a bomb, which will then attempt to reach the specified target. The Release Bomb on Laser Spot task requires that another entity lase the target.

The Release Bomb on Laser Spot task steers the bomb towards the laser point. You must synchronize the laser code for the bomber with the laser code for the lasing entity. Use of lasers is described in:

- "9.4.2 Lasing Targets" on page 304.
- "31.21. Lase Target" on page 637.
- "40.46. Laser Code" on page 885.
- "40.83. Synchronize Laser Code" on page 903.

The Release Bomb task causes a bomb to be immediately dropped. The task assumes that the target is in front of the aircraft. The targetable area is a circle whose radius is defined by the altitude of the airplane and its maximum range.

The bomb navigates toward its target based on **max-range** and **max-range-altitude** parameters (configurable in the Simulation Object Editor). A bomb can navigate **max-range**

meters laterally if it is at **max-range-altitude** meters above the target. As the altitude changes, the distance it can travel changes accordingly, so at half of **max-range-altitude** it can travel half of **max-range**.

The bomb follows a straight line path from its launch to the point it is navigating to, assuming it can reach it. If its target point is out of range, it gets as close to the target as it can. If the target point moves, the bomb changes course and heads for the new target point.

Use of a Release Bomb task requires that you determine whether the airplane is in range of the target. The Execute Surface Attack task ensures that the airplane is within range of the target before dropping its bomb. Therefore, we recommend using that task rather than one of the Release Bomb tasks. See "31.10. Execute Surface Attack" on page 630.

➤ **To assign a release bomb task:**

1. Select the entity that will release the bomb.

   If you intend to assign the Release Bomb on Laser Spot task, be sure you have synchronized the laser code as discussed earlier.
2. Choose **Task > Engagement > Release Bomb [on Laser Spot]**. The appropriate parameters display.
3. In the Bomb Types list, select the type of bomb you want to drop.
4. In the Detonation Proximity box, enter the distance from the target at which the bomb should detonate.
5. For the Release Bomb task, specify the target:
   - Click the location or enter the coordinates.
   - Select the simulation object.
6. Click OK.

31.33. Release Unguided Bomb

Fixed-wing entities equipped with the US Heavy Bomber Bomb Bay system can drop unguided (dumb bombs).

➤ **To assign a Release Unguided Bomb task:**

1. Select the entity that will release the bomb.
2. Choose **Task > Engagement > Release Unguided Bomb**. The Release Unguided Bomb parameters display.
3. In the Bomb Types list, select the type of bomb you want to drop.
4. In the Detonation Proximity box, enter the distance from the target at which the bomb should detonate.
5. Click OK.

31.34. Throw Grenade Tasks

Infantry characters can throw hand grenades at a target or at a location. An entity can throw available grenades, such as those included with VR-Forces:

- M18 - colored smoke.
- M67 - fragmentation.
- M7A3 - CS tear gas.
- M83 - white smoke.
- M84 - stun, or flashbang.

31.34.1 Throw Grenade at Location

➤ **To assign a Throw Grenade at Location task:**

1. Select the entity.
2. Choose **Task > Engagement > Throw Grenade at Location**. The Throw Grenade at Location parameters display.
3. In the Grenade Type list, select the type of grenade to throw.
4. Specify the location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
5. Click OK.

31.34.2 Throw Grenade at Target

➤ **To assign a Throw Grenade at Target task:**

1. Select the entity.
2. Choose **Task > Engagement > Throw Grenade at Target**. The Throw Grenade at Target parameters display.
3. In the Grenade Type list, select the type of grenade to throw.
4. Specify the target.
5. Click OK.



32. Human and Crowd Tasks

This chapter describes the tasks for humans and crowds that apply to entity-level scenarios. Generic movement and engagement tasks that also apply to humans are described in "Chapter 30. Movement Tasks for Entity-Level Scenarios" on page 583 and "Chapter 31. Engagement Tasks for Entity-Level Scenarios" on page 623.

32.1. Circle Around Location 650
32.2. Civilian Idle 650
32.3. Civilian Pray 650
32.4. Civilian Protest 651
32.5. Civilian Talk 651
32.6. Close Door 651
32.7. Close Window 651
32.8. Create More Entities 652
32.9. Crowd Along Line 652
32.10. Crowd Around 652
32.11. Crowd Around Waypoints 653
32.12. Crowd in Front of Entity 654
32.13. DI-Guy Animation 654
32.14. DI-Guy Gesture 654
32.15. Disperse Crowd 655
32.16. Find Cover 655
32.17. Find Firing Position 656
32.18. Fire Team Bound to Location 657
32.19. Fire Team Occupy Fire Position 658
32.20. Fire Team Travel to Waypoint 658
32.21. Flee from Entities 659
32.22. Move Between Cover to Location 659
32.23. Open Door 660
32.24. Open Window 660
32.25. Parachute to Target 660

32.26. Protest Along Line 661
32.27. Protest Around Location 662
32.28. Riot 662
32.29. Rush to Location 662
32.30. Squad Travel to Waypoint 663
32.31. Visit Neighbor 663
32.32. Wait in Queue 664
32.33. Wander 664
32.34. Wander In Area 665

32.1. Circle Around Location

The Circle Around Location task causes the members of a crowd or unit to form a circle around a location. They can face in (towards the location) or out (away from the location). This task only applies to the individually simulated members of a crowd (that is, leaf members). If a crowd has subordinate units, the task is not passed down to them.

If a crowd or unit has many members, using a small radius may result in overlapping entities.

➤ **To assign a Circle Around Location task:**

1. Select the crowd or unit.
2. Choose **Task > Movement > Circle Around Location**. The Circle Around Location parameters display.
3. Specify the location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Specify the direction to face.
5. Specify the radius of the circle to create.

32.2. Civilian Idle

The Civilian Idle task causes a human character to stand and cycle through DI-Guy animations typical of a person standing alone or in a group.

➤ **To assign a Civilian Idle task:**

1. Select a human character.
2. Choose **Task > Movement > Civilian Idle**.

32.3. Civilian Pray

The Civilian Pray task causes a human character to perform a praying animation. The entity performs the animation until something interrupts it.

➤ To assign a Civilian Pray task, select a human character and then choose **Task > Movement > Civilian Pray**.

32.4. Civilian Protest

The Civilian Protest task causes a human character to stand and cycle through DI-Guy animations typical of a person participating in a protest.

> If you need many people to protest, use the Protest Along Line or Protest Around Location crowd tasks rather than assigning this task to individuals. See "32.26. Protest Along Line" on page 661 or "32.27. Protest Around Location" on page 662.

➤ To assign a Civilian Protest task, select a human character and then choose **Task > Movement > Civilian Protest**.

32.5. Civilian Talk

The Civilian Talk task causes a human character to stand and cycle through DI-Guy animations typical of a person engaging in a conversation.

➤ To assign a Civilian Talk task, select a human character and then choose **Task > Movement > Civilian Talk**.

32.6. Close Door

If a human is within approximately one meter of an open door, this task causes the door to close. This task applies to doors controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing doors, see the chapter on *Dynamic Terrain in VR-Forces* in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To assign a Close Door task:**

1. Move a human to within one meter of an open door.
2. Choose **Task > Terrain > Close Door**.

32.7. Close Window

If a human is within approximately one meter of an open window, this task causes the window to close. This task applies to windows controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing windows, see the chapter on *Dynamic Terrain in VR-Forces* in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To assign a Close Window task:**

1. Move a human to within one meter of an open window.
2. Choose **Task > Terrain > Close Window**.

32.8. Create More Entities

The Create More Entities task adds more entities to a crowd or herd. If the crowd or herd is part of a pedestrian or animal area, the new entities wander within the area. If the crowd or herd is not part of an area, their wandering is unrestricted.

When the Create More Entities task runs, it gives each new entity a task. It assigns humans a Move To Location task, followed by a Wander task. It assigns animals a Wander With Animations task. The Create More Entities task is not complete until all of the Wander tasks have been sent. Therefore, the task does not complete as quickly as you might expect simply based on when the entities are created.

➤ **To assign a Create More Entities task:**

1. Select a crowd or a herd, or select a pedestrian or herd area.
2. Choose **Task > Create More Entities**. The Create Pedestrians parameters display.
3. Specify the number of entities to create.
4. Select a placement option.
5. Click OK.

32.9. Crowd Along Line

The Crowd Along Line task causes a crowd to gather to the left or right of a line at its center point.

➤ **To assign a Crowd Along Line task:**

1. Select a crowd.
2. Choose **Task > Movement > Crowd Along Line**. The Crowd Along Line parameters display.
3. Select the line the crowd should crowd along.
4. Specify left or right of the line.
5. Click OK.

32.10. Crowd Around

The Crowd Around task causes a crowd unit to cluster around the specified location, simulation object, or single point tactical graphic.

- The Crowd Around task calculates target locations for the members of the crowd to move to that are greater than or equal to the minimum distance specified from the object. However, due to interactions between the crowd members as they move, it is possible that some will end up closer to the target object than the minimum distance.
- The Crowd Around Waypoints task causes a crowd to gather around one of potentially multiple waypoints and then perform a behavior for a specified length of time. See "32.11. Crowd Around Waypoints" below.

➤ **To assign a Crowd Around task:**

1. Select a crowd.
2. Choose **Task > Movement > Crowd Around**. The Crowd Around parameters display.
3. Specify where the crowd should gather:
   - Click the location to move to or enter the coordinates.
   - Select the simulation object or point tactical graphic.
4. In the Minimum Distance box, specify the minimum distance between a member of the crowd and the specified location, entity, or point. Some members of the crowd may stand further back, ensuring that they do not collide with others in the crowd.
5. Click OK.

32.11. Crowd Around Waypoints

The Crowd Around Waypoints task causes members of a crowd to gather at one or more waypoint and perform the selected behavior for a specified amount of time. If you specify multiple waypoints, the individual crowd members move to the waypoint closest to them.

➤ **To assign a Crowd Around Waypoints task:**

1. Select a crowd.
2. Choose **Task > Movement > Crowd Around Waypoints**. The Crowd Around Waypoints parameters display.
3. Select one or more waypoints where the crowd should gather.
4. In the Behavior list, select Talk or Pray to indicate what the members of the crowd should do at the waypoint.
5. In the Duration box, enter the number of minutes that the crowd should perform the selected behavior. Once the time has elapsed, the crowd is ready to perform another task. They continue performing the behavior until another task is assigned.
6. In the Minimum Distance box, specify the minimum distance between a member of the crowd and the waypoint. Some members of the crowd may stand further back, ensuring that they do not collide with others in the crowd.
7. Click OK.

32.12. Crowd in Front of Entity

The Crowd in Front of Entity task causes a crowd to gather in front of an entity, where the front of the entity is determined based on its heading.

➤ **To assign a Crowd in Front of Entity task:**

1. Select a crowd.
2. Choose **Task > Movement > Crowd in Front of Entity**. The Crowd in Front of Entity parameters display.
3. Select the entity you want the crowd to gather in front of.
4. Click OK.

32.13. DI-Guy Animation

VR-Forces can specify the movements and appearance of DI-Guy models used in the Stealth observer mode and in 3D visualization programs such as VR-Vantage Stealth. The DI-Guy Animation task directs the 3D model to engage in a one-time or repeated series of motions. VR-Forces does not simulate the actions that the 3D model performs; it just adjusts the location and orientation of the VR-Forces entity to synchronize it with the 3D model's animated behavior. To change a 3D model's appearance, use the DI-Guy Appearance set data request. For details, see "40.25. DI-Guy Characteristics (Appearance, Head, Body Weight)" on page 876.

ⓘ The DI-Guy Animation task applies only to lifeform entities.

➤ **To assign a DI-Guy Animation task:**

1. Select the lifeform entity whose behavior you want to animate.
2. Choose **Task > Movement > DI-Guy Animation**. The DI-Guy Animation parameters display.
3. From the DI-Guy Animation list, select an animation.
4. In the Animate group box, specify the duration of the animation.
5. Optionally, select Preview to see the entity perform the animation in the 3D (Stealth) view before you assign the task.
6. Click OK.

32.14. DI-Guy Gesture

The DI-Guy Gesture task lets you assign a gesture to a human character. Gestures are motions that a character makes using part of its body. By contrast, animations may involve moving from one place to another.

➤ **To assign a DI-Guy Gesture task:**

1. Select the lifeform entity whose behavior you want to animate.
2. Choose **Task > Movement > DI-Guy Gesture**. The DI-Guy Gesture parameters display.
3. In the Gestures list, select a gesture.
4. Select the Wait Until Complete check box if you do not want the entity to start another gesture until this one is finished.
5. Optionally, select Preview to see the entity perform the animation in the 3D (Stealth) view before you assign the task.
6. Click OK.

32.15. Disperse Crowd

The Disperse Crowd task causes a crowd to wander freely. If the crowd was created as part of a pedestrian area, this task allows it to wander outside the area. (A Wander task would restrict it to the area.) The primary purpose of this task is to disperse a crowd that has previously crowded around an entity or location.

➤ **To assign a Disperse Crowd task:**

1. Select the crowd you want to disperse.
2. Choose **Task > Movement > Disperse Crowd**.

32.16. Find Cover

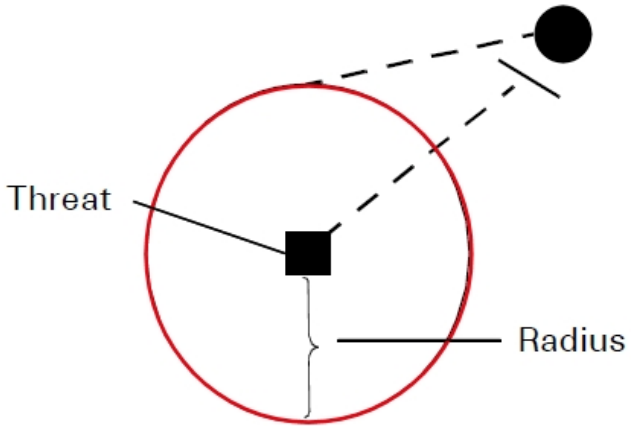
The Find Cover task causes a human entity to find a location from which a specified entity (the threat) cannot see it. The task takes a range, a distance from threat, and threat radius. The Find Cover task requires that the entity be in a navigation area.

The range is the distance from its starting location within which the tasked entity will seek cover. The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

The threat radius defines an area around the threat from which line-of-sight calculations are made. Figure 241 illustrates this concept. Line-of-sight between the threat and the covered entity is blocked. However, the covered entity is still visible from a point within the threat radius.

ⓘ The entity seeking cover cannot hide behind other entities. It only hides behind terrain features, such as walls or doors.

Figure 241. Threat radius



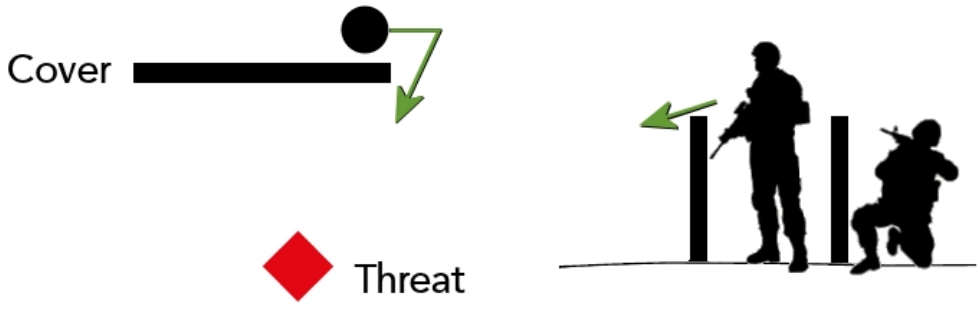
➤ **To assign a Find Cover task:**

1. Select an entity.
2. Choose **Task > Movement > Find Cover**. The Find Cover parameters display.
3. Select the entity to find cover from (the threat).
4. Specify the Threat Radius.
5. Specify the Range.
6. Specify the Distance From Threat.
7. Click OK.

32.17. Find Firing Position

The Find Firing Position task causes a human entity to find a location from which a specified entity (the threat) cannot see it, but from which it can easily fire on the threat entity. Figure 242 illustrates this concept. The entity under cover can easily move to a location that has line-of-sight to the threat. This could be by moving to the side or looking over the top of the terrain feature it is hiding behind. The Find Firing Position task requires that the entity be in a navigation area.

Figure 242. Firing position



> The entity seeking cover cannot hide behind other entities. It only hides behind terrain features, such as walls or doors.

The task takes a range, a distance from threat, and a threat radius value. The range is the distance from its starting position within which the tasked entity will seek cover. The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

The threat radius defines an area around the threat from which line-of-sight calculations are made. Figure 241 on the previous page illustrates this concept. Line-of-sight between the threat and the covered entity is blocked. However, the covered entity is still visible from a point within the threat radius.

➤ **To assign a Find Firing Position task:**

1. Select the entity.
2. Choose **Task > Movement > Find Firing Position**. The Find Firing Position parameters display.
3. Select the entity to find cover from (the threat).
4. Specify the Threat Radius.
5. Specify the Range.
6. Specify the Distance From Threat.
7. Click OK.

32.18. Fire Team Bound to Location

The Fire Team Bound to Location task causes a fire team to move in successive bounds to the specified destination while watching the location of a threat.

Initially, the team moves to points that they can use as fire positions while hidden from the threat location. Next, the team splits into two groups and each group then alternates between watching the threat (overwatch) and moving toward the threat. While moving toward the destination, the entities rush forward approximately 10-15 meters and then assume new overwatch positions. If you select the Always Fire check box, then the entities in the overwatch position will fire toward the threat location if they have line of sight.

➤ **To assign the Fire Team Bound to Location task:**

1. Select the fire team.
2. Choose **Task > Movement > Fire Team Bound to Location**. The Fire Team Bound to Location parameters display.
3. Click on the terrain where you want the fire team to move towards, or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.

4. Specify the location of the threat, and optionally the altitude, as described in "14.4. Specifying an Object's Altitude" on page 386. The altitude can be useful if you want to specify a point above the terrain that is more visible to the fire team.
5. If you want the members of the team to fire at the threat location while they are in the overwatch position, select the Always Fire check box.
6. Click OK.

32.19. Fire Team Occupy Fire Position

The Fire Team Occupy Fire Position task causes a unit to find firing positions. This is similar to the Find Firing Position task, but it works for a unit instead of individuals. See "32.17. Find Firing Position" on page 656 for details.

➤ **To assign the Fire Team Occupy Fire Position task:**

1. Select the unit.
2. Choose **Task > Movement > Fire Team Occupy Fire Position**. The Fire Team Occupy Fire Position parameters display.
3. Specify the Fire Position.
4. Specify the Range.
5. Select the entity to find cover from (the threat).
6. Specify the Threat Radius.
7. Specify the Distance From Threat.
8. Click OK.

32.20. Fire Team Travel to Waypoint

You can order a fire team to move to a waypoint in a specified formation. The formations and their spacing are defined in the Simulation Object Editor. You can override the spacing when you assign the task.

To learn more about configuring formations, refer to "71.5. Editing Formations" on page 1411.

➤ **To assign the Fire Team Travel to Waypoint task:**

1. Select the fire team.
2. Choose **Task > Movement > Fire Team Travel to Waypoint**. The Fire Team Travel to Waypoint parameters display.
3. In the Name box, specify the name of the waypoint to move to.
4. From the Formation list, select the formation that the fire team should move in.
5. Optionally, select the Override Formation Spacing check box to have the entities in the team move with a different spacing than is specified in the SMS.

6. If you chose to override the formation spacing, enter the new spacing in the Minimum Entity Spacing box.
7. Click OK.

32.21. Flee from Entities

The Flee from Entities task can be assigned to individual humans and to crowds. It causes the human or crowd to flee from one or more specified entities until it is a specified distance away. The entities must be within a navigation area.

The version of the task for individual humans has an option, Allow Destroyed, which causes the entity continue to flee from the threat even if it is destroyed. In that case, the entity continues to flee to the specified distance. If you do not select that option, the Flee from Entities task ends if the threat is destroyed. Note that the entity might continue to run if it is within the Flee from Threat range when that reactive task is enabled. See "34.14. Reactive Tasks" on page 707.

➤ **To assign a Flee from Entities task:**

1. Select the entity, either a individual human or crowd.
2. Choose **Task > Movement > Flee From Entities**. The Flee From Entities parameters display.
3. Select each entity that you want the individual human or crowd members to flee from.
4. Specify the radius of the flight zone.
5. For an individual human, you can optionally select the Allow Destroyed check box to have the entity continue to flee from the threat after it is destroyed.
6. Click OK.

32.22. Move Between Cover to Location

The Move Between Cover to Location task causes an entity to move to a location by moving between cover points (locations where it is not visible to a specified threat entity). The task requires you to specify a distance from the threat entity and a threat radius.

> The entity seeking cover cannot hide behind other entities. It only hides behind terrain features, such as walls or doors.

The distance from threat is the minimum distance to maintain between the threat and the tasked entity. For example, you wouldn't expect the entity to run right next to the threat to reach some point on the other side of it. You would expect it to maintain some distance from the threat entity.

VR-Forces calculates cover points based on line-of-sight from a threat radius around the threat entity. Figure 241 illustrates this concept. Line-of-sight between the threat and the covered entity is blocked. However, the covered entity is still visible from a point within the threat radius.

> The default values for threat radius and distance from threat are fairly small. Unless you are playing in a congested area, you will probably want to change them.

➤ **To assign a Move Between Cover to Location task:**

1. Select the entity.
2. Choose **Task > Movement > Move Between Cover to Location**. The Move Between Cover to Location parameters display.
3. Click on the terrain to specify the location, or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Select the Threat entity.
5. In the Threat Radius box, specify the threat radius.
6. In the Distance From Threat box, specify the minimum distance to maintain between the threat entity and the moving entity.
7. Click OK.

32.23. Open Door

If a human is within approximately one meter of a closed door, this task causes the door to open. This task applies to doors controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing doors, see the chapter on *Dynamic Terrain in VR-Forces* in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To assign an Open Door task:**

1. Move a human to within one meter of a closed door.
2. Choose **Task > Terrain > Open Door**.

32.24. Open Window

If a human is within approximately one meter of a closed window, this task causes the window to open. This task applies to windows controlled by switch nodes in the terrain, not articulated parts. For more information about opening and closing windows, see the chapter on *Dynamic Terrain in VR-Forces* in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To assign an Open Window task:**

1. Move a human to within one meter of a closed window.
2. Choose **Task > Terrain > Open Window**.

32.25. Parachute to Target

The Parachute to Target task causes a paratrooper entity to disembark from an aircraft, deploy its parachute, and try to land at the target object or location. The paratrooper tries to land as close to the target as it can given the altitude at which it disembarks and the

distance to the target. If you specify a muster point, the paratrooper walks to that point after landing.

- While an entity is parachuting, if you double-click that entity to try to move it, it snaps to the terrain.
- The Air Drop task can also cause paratroopers to parachute. For details, see "30.1. Airdrop Along Route" on page 585.
- Aircraft can task multiple paratroopers to parachute to a target and a muster point. For details, see "30.35. Paratrooper Air Drop" on page 608.

➤ **To assign a Parachute to Target task:**

1. Select the paratrooper.
2. Choose **Task > Movement > Parachute to Target**. The Parachute to Target parameters display.
3. Specify a target point where the paratrooper will try to land.
4. Optionally, select Use Muster Point and then specify where the paratrooper should move after it lands.
5. Specify which criterion the entity should use to determine when to deploy its parachute.
   - Select Deploy Parachute at Altitude and enter the altitude at which to deploy the parachute. By default, Deploy Altitude is AGL is selected to specify that the deployment altitude is above ground level instead of mean sea level (MSL). Clear the check box to use MSL instead.

     If the aircraft is below the deployment altitude, the parachute deploys as soon as the entity disembarks.
   - Select Deploy at Elapsed Time and then enter the Elapsed Time to Deploy. This is the length of time that the paratrooper waits to deploy the parachute after it begins to fall.
6. Click OK.

32.26. Protest Along Line

The Protest Along Line task causes a crowd to gather along a specified line and continuously execute a series of protest animations.

➤ **To assign a Protest Along Line task:**

1. Select the crowd.
2. Choose **Task > Movement > Protest Along Line**. The Protest Along Line parameters display.
3. Specify a line.
4. Specify which side of the line to protest on.
5. Click OK.

32.27. Protest Around Location

The Protest Around Location task causes a crowd to gather around a specified location and continuously execute a series of protest animations.

➤ **To assign a Protest task:**

1. Select the crowd.
2. Choose **Task > Movement > Protest**. The Protest Around Location parameters display.
3. Specify a location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Click OK.

32.28. Riot

The Riot task causes a crowd to move into an area specified for the riot, going to Riot points or other places in the area. The members of the crowd run and perform angry gestures. You can also specify that the rioters should start fires, which are ignited at each riot point inside the area. Rioters ignore riot points placed outside the area.

The area where the riot occurs can be any Area, while Riot Points are a specific type of simulation effect object.

➤ **To assign a Riot task:**

1. Select a crowd.
2. Choose **Task > Movement > Riot**. The Riot parameters display.
3. Select an area where the riot will occur.
4. Optionally, select the Start Fires check box. Rioters will start fires at the riot points inside the specified area.
5. Click OK.

32.29. Rush to Location

The Rush to Location task causes an entity to move quickly to the specified location and then assume a prone position or, optionally, a kneeling position.

➤ **To assign a Rush to Location task:**

1. Select an entity.
2. Choose **Task > Movement > Rush to Location**. The Rush to Location parameters display.
3. Specify a location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Specify the location of the threat.

5. Optionally, select the Destination Is High Cover check box. This causes the entity to kneel at the specified location rather than lying prone.
6. Click OK.

32.30. Squad Travel to Waypoint

You can order a squad to move to a waypoint. The squad's formation is a lead fire team that moves in a wedge formation, followed by the squad leader and then the trailing fire team. The spacing of the formation is defined in the Simulation Object Editor but you can override it when you assign the task.

To learn more about configuring formations, refer to "71.5. Editing Formations" on page 1411.

If you select the Use Overwatch check box, then the squad leader and the trailing fire team follow the lead fire team at a greater distance. Another waypoint is created just past the waypoint specified in the task. The lead fire team gathers at this new waypoint, while the squad leader and the trailing fire team gather at the task's waypoint.

➤ **To assign the Squad Travel to Waypoint task:**

1. Select the squad.
2. Choose **Task > Movement > Squad Travel to Waypoint**. The Squad Travel to Waypoint parameters display.
3. In the Destination box, enter the name of the waypoint to move to. You can select it from a list, on the terrain, or enter the name.
4. Optionally, select the Use Overwatch check box to increase the following distance between the lead fire team and the squad leader and the trailing fire team.
5. Optionally, select the Override Formation Spacing check box to have the entities in each fire team move with a formation spacing that is different than is specified in the SMS.
6. If you chose to override the formation spacing, enter the new spacing in the Minimum Entity Spacing box, in meters.
7. Click OK.

32.31. Visit Neighbor

Use the Visit Neighbor task to have a human entity move to a nearby human or ground entity.

➤ **To assign the Visit Neighbor task:**

1. Select the human entity.
2. Choose **Task > Movement > Visit Neighbor**. The Visit Neighbor parameters display.
3. Enter the Radius.
4. Click OK.

32.32. Wait in Queue

The Wait in Queue task causes a human to move to the beginning of a Queue simulation effect object and then move along the line until it stands behind the last human in the line. (The first human in line moves to the end point of the simulation effect object, which is the beginning of the queue.) As people leave the top of the queue, the entity moves up in line.

Since this is a Wait task, the entity does not move to the next task in its plan until it is released from the queue. Entities are released from a queue when the Release Next Entity From Queue task is sent to the queue. For more information, see "34.12. Release Next Entity From Queue" on page 706.

> The Wait in Queue task only works with a Queue simulation effect object. It does not work with other types of lines.

➤ **To assign a Wait in Queue task:**

1. Select the entity.
2. Choose **Task > Wait > Wait in Queue**. The Wait in Queue parameters display.
3. Select the Queue that the entity should wait in.
4. Click OK.

32.33. Wander

The Wander task causes a lifeform entity to move to a series of randomly chosen locations for the specified amount of time. You can constrain wandering to a specified area. If the entity is using a preferred navigation mode and there is preferred road data for the terrain, the Wander task will only choose destinations that meet the preference criteria. (For information about setting navigation preferences, see "40.54. Navigation Preferences" on page 890.) The entity must be in a navigation area.

➤ **To assign a Wander task:**

1. Select the entity.
2. Choose **Task > Movement > Wander**. The Wander parameters display.
3. Select a Duration option: Indefinitely or Timed. If you select Timed, specify the amount of time to wander in days, hours, minutes, and seconds.
4. Select a Movement option: Free or Restricted To Area. If you select Restricted To Area, enter the name of the area where you want the entity to wander.
5. From the Destination Restriction list, choose an option for how the entity will move.
6. Click OK.

32.34. Wander In Area

The Wander In Area task causes a crowd to wander in the specified area. The entity must be in a navigation area.

➤ **To assign a Wander In Area task:**

1. Select the crowd.
2. Choose **Task > Movement > Wander In Area**. The Wander In Area parameters display.
3. Select the area to wander in.
4. Click OK.



33. Unit Behavior Tasks for Entity-Level Scenarios

This chapter describes the complex tasks for units that apply to entity-level scenarios. These tasks involve a number of subtasks and are more complex than the tasks covered in other chapters.

33.1. Overview of Unit Behaviors 668
33.2. Drop Off Passengers at Location 669
33.3. Pick Up Passengers at Location 669
33.4. Tasks for Mounting and Dismounting Units 669
33.4.1 Dismount Headquarters 669
33.4.2 Dismount Platoon 670
33.4.3 Dismount Squad 670
33.4.4 Dismount Team 671
33.4.5 Mount Headquarters 671
33.4.6 Mount Platoon 671
33.4.7 Mount Squad 672
33.4.8 Mount Team 672
33.5. Movement Tasks for Unit Behaviors 673
33.5.1 Bound In Lanes 673
33.5.2 Bounding Overwatch Successive Continuous 673
33.5.3 Follow Along Offset Route 673
33.6. Defensive Tasks for Unit Behaviors 674
33.6.1 Assume Coil Formation 674
33.6.2 Provide Mobile Overwatch 674
33.6.3 Provide Stationary Overwatch 675
33.7. Offensive Tasks for Unit Behaviors 675
33.7.1 Assault 675
33.7.2 Attack by Fire 676
33.7.3 Attack By Fire in Rotation 676
33.7.4 Clear 676
33.7.5 Engage From Vehicle/Soldier Firing Positions 677

33.7.6 Movement To Contact 677
33.7.7 Seize Objective 678
33.7.8 Suppressive Fire 678

33.1. Overview of Unit Behaviors

VR-Forces supports complex unit behaviors using the VR-Forces Lua Behavior Engine. Military unit behaviors often depend on the terrain. Having an armored company seize an objective in a desert is radically different than having a company seize an objective in a dense city or a forest. Things such as roads, rivers, railroads, bridges, and other natural and man-made obstacles can have a significant impact on how a task is performed.

MAK has designed these behaviors using open source United States doctrine to work in an open terrain such as the U.S. National Training Center at Fort Irwin. This is a fairly empty terrain designed for desert warfare. While the tasks may work well in other similar terrains, your results may vary. We do not support all tasks in all terrains. The VR-Forces behavior engine allows for a tremendous amount of complexity in behaviors, and we believe some modification to these behaviors may be needed for any deployed system. The scripts used to design these tasks are provided in VR-Forces for you to adjust and update to meet your doctrinal and terrain goals. Fully understanding how to extend these behaviors is complicated and MAK recommends developer training to best meet complex behavior needs. Please contact us at sales@mak.com so that we can help you further understand your options.

There are some concepts you should understand to use these tasks most effectively.

- Vehicle Crew. A vehicle crew is a notional subordinate of unit defined with the Vehicle Crew function. See "71.4.6 Defining Functions for Subordinates" on page 1410. Tasks for crews require that the unit knows what vehicles are associated with it, and so require that the unit has a vehicle crew in its unit composition.
- Base of fire and maneuver elements. When using an overwatch defense task, you specify one or more maneuver elements. These are the members of a unit who will move and shoot.

There are several Behavior Ground scenarios in *./userData/scenarios/Sample/VR-TheWorld_Online* that include several vignettes that demonstrate some of the unit behaviors.

Field artillery units in entity-level scenarios are aware of the subordinates and the weapons in the unit, knowing their capabilities and remaining munition rounds. This allows the unit to determine which subordinates can be tasked in an assault.

33.2. Drop Off Passengers at Location

Use the Drop Off Passengers at Location task to instruct a vehicle crew to drop off their squad at a given location.

➤ **To task a vehicle crew to drop off the squad:**

1. Select the crew.
2. Choose **Task > Embarkation > Drop Off Passengers at Location**. The Drop Off Passengers at Location parameters display.
3. Specify the drop off location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. In the Objects List, select the passengers to drop off. They are added to the Passengers list.
5. Enter the heading that the crew should face when they arrive at the drop off location.
6. Click OK.

33.3. Pick Up Passengers at Location

Use the Drop Off Passengers at Location task to instruct a vehicle crew to pick up their squad at a given location.

➤ **To task a vehicle crew to drop off the squad:**

1. Select the unit's vehicle crew.
2. Choose **Task > Embarkation > Pick Up Passengers at Location**. The Pick Up Passengers at Location parameters display.
3. Specify the pickup location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Enter the heading that the crew should face when they arrive at the pickup location.
5. Click OK.

33.4. Tasks for Mounting and Dismounting Units

There are different tasks for mounting and dismounting units from vehicles. The one you should use depends on the type of unit.

33.4.1 Dismount Headquarters

Use the Dismount Headquarters task to tell an infantry platoon headquarters to exit a vehicle and gather at a rally point. Optionally, you can specify whether the platoon sergeant or other habitually mounted entities remain on board.

➤ **To task a headquarters section to dismount a vehicle:**

1. Select the unit.
2. Choose **Task > Embarkation > Dismount Headquarters**. The Dismount Headquarters parameters display.
3. Specify the location of the rally point. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Optionally, select or clear the Retain Habitual Mountees check box.
5. Click OK.

33.4.2 Dismount Platoon

Use the Dismount Platoon task to tell an infantry platoon to exit a vehicle and gather in formation around a point.

➤ **To task a platoon unit to dismount a vehicle:**

1. Select the platoon.
2. Choose **Task > Embarkation > Dismount Platoon**. The Dismount Platoon parameters display.
3. For the Dismount Point, specify the waypoint.
4. Enter the direction that the platoon should face. Also specify whether the direction is relative to True or Magnetic north.
5. Select the Vehicle Formation from the list.
6. From the Threat Level list, selected the expected threat from any nearby enemies. This determines the posture that the dismounted subunits will assume.
7. Optionally, select which subunits should dismount. Unselected units remain on the vehicle.
8. Click OK.

33.4.3 Dismount Squad

The Mount Headquarters task causes an infantry squad to exit a vehicle and gather at a rally point.

➤ **To task a squad unit to dismount a vehicle:**

1. Select the squad.
2. Choose **Task > Embarkation > Dismount Squad**. The Dismount Squad parameters only.
3. From the Threat Level list, selected the expected threat from any nearby enemies. This determines the posture that the dismounted entities will assume.
4. Optionally, select the Dismount Crew check box to also dismount the vehicle team.
5. Click OK.

33.4.4 Dismount Team

Use the Dismount Team task to tell an infantry team to exit a vehicle and gather at a rally point.

➤ **To task a team unit to dismount a vehicle:**

1. Select the team.
2. Choose **Task > Embarkation > Dismount Team**. The Dismount Team parameters display.
3. Specify the coordinates of the rally point or click a point on the terrain.
4. Click OK.

33.4.5 Mount Headquarters

The Mount Headquarters task causes an infantry platoon headquarters section to gather around a rally point and then board a vehicle. The vehicle is the one specified by the Assign Transport set data request. See "40.96.1 Assign Transport" on page 909.

➤ **To task a headquarters section to mount a vehicle:**

1. Select the HQ section.
2. Choose **Task > Embarkation > Mount Headquarters**. The Mount Headquarters parameters display.
3. Specify the location of the rally point. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. In the Transports List or the window, select the entities to mount.
5. Click OK.

33.4.6 Mount Platoon

The Mount Platoon task causes an infantry platoon unit to gather around a point and then board a vehicle. The vehicle is the one specified by the Assign Transport set data request. See "40.96.1 Assign Transport" on page 909.

➤ **To task a platoon to mount a vehicle:**

1. Select the platoon.
2. Choose **Task > Embarkation > Mount Platoon**. The Mount Platoon parameters display.
3. For the Mount Point, specify a point.
4. Enter the direction that the platoon should face when they arrive at the mount point. Additionally, specify whether the direction is relative to True or Magnetic north.
5. Select the Vehicle Formation from the list. The platoon assumes this formation.
6. From the Threat Level list, selected the expected threat from any nearby enemies. This determines the posture that the dismounted subunits will assume.

7. Optionally, select which subunits should mount the vehicle. Unselected units remain in formation at the Mount Point.
8. Click OK.

33.4.7 Mount Squad

The Mount Squad task causes an infantry squad to board a vehicle. You can choose to have the squad form up at a rally point before boarding the vehicle. The vehicle is the one specified by the Assign Transport set data request; see "40.96.1 Assign Transport" on page 909. When tasking the squad, you can assign a different vehicle.

➤ **To task a squad to mount a vehicle:**

1. Select the squad unit.
2. Choose **Task > Embarkation > Mount Squad**. The Mount Squad parameters display.
3. Optionally, select the Reassign Transport check box and then specify a vehicle.
4. Specify the location of the rally point. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
5. Enter the direction that the squad should face. Additionally, specify whether the direction is relative to True or Magnetic north.
6. Select the Formation from the list.
7. From the Threat Level list, selected the expected threat from any nearby enemies. This determines the posture that the dismounted subunits will assume.
8. Click OK.

33.4.8 Mount Team

The Mount Team task causes an infantry team to board a vehicle. You can choose to have the team form up at a rally point before boarding the vehicle. The vehicle is the one specified by the Assign Transport set data request; see "40.96.1 Assign Transport" on page 909. When tasking the team, you can assign a different vehicle.

➤ **To task a team to mount a vehicle:**

1. Select the team unit.
2. Choose **Task > Embarkation > Mount Team**. The Mount Team parameters display.
3. Optionally, select the Reassign Transport check box and then specify a vehicle.
4. Optionally, specify the coordinates of the rally point or click a point on the terrain.
5. Click OK.

33.5. Movement Tasks for Unit Behaviors

Units can execute different movement tasks.

33.5.1 Bound In Lanes

The Bound In Lanes task causes a fire team to move in parallel lanes, in successive bounds, to assault a threat location.

➤ **To assign the Bound In Lanes task:**

1. Select the fire team.
2. Choose **Task > Movement > Bound In Lanes**. The Bound In Lanes parameters display.
3. Choose the Enemy Area.
4. Specify the Start Point for the movement. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
5. Click OK.

33.5.2 Bounding Overwatch Successive Continuous

The Bounding Overwatch Successive task causes a mechanized ground unit to move along a route in successive bounds. They will continue along the route, regardless of whether they contact opposing forces. You specify the minimum and maximum bounding lengths, as well as the possible threat locations that the unit should watch.

➤ **To assign the Bounding Overwatch Successive Continuous task:**

1. Select the platoon.
2. Choose **Task > Movement > Bounding Overwatch Successive Continuous**. The Bounding Overwatch Successive Continuous parameters display.
3. Select the Travel Route.
4. Enter a Minimum Bound Length.
5. Enter a Maximum Bound Length.
6. Select one or more Enemy Locations, which are area tactical graphics.
7. Click OK.

33.5.3 Follow Along Offset Route

The Follow Along Offset Route task causes one ground entity follow another at a specified offset. The leader is expected to be tasked to move. If you tell two or more entities to follow the same simulation object, make sure that you do not give them the same offset (see "42.5.5 Following Simulation Objects" on page 945).

➤ **To assign the Follow Along Offset Route task:**

1. Select the entity that will follow another.
2. Choose **Task > Movement > Follow Along Offset Route**. The Follow Along Offset Route parameters display.
3. Select the Leader Route.
4. Select the Leader.
5. Enter the offset parameters.
6. Click OK.

33.6. Defensive Tasks for Unit Behaviors

Units can execute different defense tasks.

33.6.1 Assume Coil Formation

The Assume Coil Formation task causes a mechanized ground unit to move into a defensive coil, facing outward and distanced from a central point at the specified radius.

➤ **To assign the Assume Coil Formation task:**

1. Select the tank platoon.
2. Choose **Task > Defense > Assume Coil Formation**. The Assume Coil Formation parameters display.
3. Specify the Radius.
4. Click OK.

33.6.2 Provide Mobile Overwatch

The Provide Mobile Overwatch task tells member of a vehicle crew to provide overwatch of an enemy area, moving with the crew's squad.

When you assign the Provide Mobile Overwatch task, there are parameters to specify:

- Maneuver Elements. The members of the unit that are moving. The vehicle associated with the unit provides the overwatch support.
- Enemy area. The area where the enemy is located.

➤ **To task a vehicle crew to provide mobile overwatch:**

1. Select the crew.
2. Choose **Task > Defense > Provide Mobile Overwatch**. The Provide Mobile Overwatch parameters display.
3. Select a Maneuver Element object. Choose one or more entities or units in the Objects List or terrain.
4. For the Enemy Area, click the Select button () and then select the area.

5. Enter the Overwatch Distance at which the base of fire element will trail the Maneuver Elements.
6. Click OK.

33.6.3 Provide Stationary Overwatch

The Provide Stationary Overwatch task tells members of a vehicle crew to provide overwatch while remaining at the overwatch location.

When you assign the Provide Stationary Overwatch task, there are parameters to specify:

- Overwatch location. The vehicle moves to this location and provides overwatch defense from there.
- Maneuver Elements. The members of the unit that are moving. The vehicle associated with the unit provides the overwatch support.
- Enemy area. The area where the enemy is located.

➤ **To task a vehicle crew to provide stationary overwatch:**

1. Select the crew.
2. Choose **Task > Defense > Provide Stationary Overwatch**. The Provide Stationary Overwatch parameters display.
3. Specify the Overwatch Location. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Select a Maneuver Element object. Choose one or more entities or units in the Objects List or terrain.
5. Optionally, select the Enemy Area.
6. Click OK.

33.7. Offensive Tasks for Unit Behaviors

Units can execute different offense tasks.

33.7.1 Assault

The Assault task causes a mechanized ground unit to assault an area without moving past a specified simulation object.

➤ **To assign the Assault task:**

1. Select the unit.
2. Choose **Task > Offense > Assault**. The Assault parameters display.
3. Select the Enemy Area.
4. Select the line where the unit will set up its Support By Fire Position.

5. Specify the location of the Assault Position. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
6. Select the simulation object to be the Limit of Advance. The unit will not move past this point.
7. Click OK.

33.7.2 Attack by Fire

The Attack By Fire task causes a mechanized ground unit to move to a line where they take position and fire at an area. The unit can fire continuously, or stop after a specified length of time when it does not detect contacts in the area.

➤ **To assign an Attack By Fire task:**

1. Select the unit.
2. Choose **Task > Offense > Attack By Fire**. The Attack By Fire parameters display.
3. Select the Line of Fire that represents the hull-down position for the tank platoon.
4. Select the Enemy Area to use as the engagement zone.
5. In the Time Out box, indicate the length of time (in seconds) that the tank platoon should wait between contacts before ending the task.
6. Select the Do Fires Indefinitely check box to have the tank platoon continue firing until their ammunition is depleted.
7. Click OK.

33.7.3 Attack By Fire in Rotation

The Attack By Fire in Rotation task causes a mechanized ground unit to move to and rotate through a battle position, taking turns firing at an area.

➤ **To assign an Attack By Fire in Rotation task:**

1. Select the unit.
2. Choose **Task > Offense > Attack By Fire in Rotation**. The Attack By Fire in Rotation parameters display.
3. Specify the Battle Position. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Select the Target Area to use as the engagement zone.
5. Click OK.

33.7.4 Clear

A mechanized ground unit tasked to perform the Clear task lines up at a starting position and then moves to a line, attempting to remove all opposing forces along the way.

➤ **To assign the Clear task:**

1. Select unit.
2. Choose **Task > Offense > Clear**. The Clear parameters display.
3. Select the line that is the Limit of Advance. Surviving members of the platoon stop when they reach the line.
4. Select the Starting Point where the platoon will assemble.
5. Click OK.

33.7.5 Engage From Vehicle/Soldier Firing Positions

Use the Engage From Vehicle (or Soldier) Firing Positions to have a mechanized ground entity move first to a hide position, pause, continue on to a fire position, and then fire on an enemy location.

➤ **To assign the Engage From Vehicle (or Soldier) Firing Positions task:**

1. Select the entity.
2. Choose **Task > Engagement > Engage From Vehicle Firing Positions** (or **Engage From Soldier Firing Positions**). The parameters display.
3. Specify the location of the Hull Down Position (or Shoot Position). You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Specify the location of the Hide Position.
5. Select the Delayed Start check box to have the entity pause and allow another entity to engage first.
6. Select the Enemy Area.
7. Click OK.

33.7.6 Movement To Contact

The Movement To Contact task causes a mechanized ground unit to move towards an objective. Upon making contact with opposing forces, the task ends with the unit performing the specified action.

➤ **To assign the Movement To Contact task:**

1. Select the unit.
2. Choose **Task > Offense > Movement To Contact**. The Movement to Contact parameters display.
3. Select the Axis of Attack, which is a line.
4. Select the Objective, which is an ellipse or circle.
5. Specify the Action on Contact: Engage or Ignore.
6. Click OK.

33.7.7 Seize Objective

The Seize Objective task causes a tank company to move to take control of an area.

➤ **To assign the Seize Objective task:**

1. Select the company.
2. Choose **Task > Offense > Seize Objective**. The Seize Objective parameters display.
3. Select the Objective Area for the company to seize.
4. Select the Fire Positions, which are lines where platoons within the company should position themselves to provide supportive fire.
5. Select one Line of Departure.
6. Select the Seize Direction: Direct, Left, or Right.
7. If the platoons within the company should make the attack, select the Two Platoon Assault check box. If one platoon should provide supporting fire while the other seizes the objective, clear the check box.
8. If there is an artillery unit that can provide support for the company, select it in the Artillery box.
9. Optionally, specify the Assault Route that the company will follow to move to the objective area.
10. Click OK.

33.7.8 Suppressive Fire

The Suppressive Fire task causes the subordinates of a mechanized ground unit to fire at a specified simulation object. If other opposing forces are detected, a subordinate engages with them. For more information, see "26.1. Suppression" on page 520.

➤ **To assign the Suppressive Fire task:**

1. Select the unit.
2. Choose **Task > Offense > Suppressive Fire**. The Suppressive Fire parameters display.
3. Use the Filter to select whether you want to target a ground entity or a point.
4. Select the target.
5. Click OK.



34. Embarkation, Wait, Radio, and Other Tasks

This chapter describes the tasks supplied with VR-Forces that are categorized as Embarkation, Radio, Wait, Other, and uncategorized.

34.1. Embarkation Tasks 681
34.1.1 Disembark 681
34.1.2 Disembark All 681
34.1.3 Embark 681
34.1.4 Airdrop 683
34.1.5 Airdrop from Aircraft 684
34.1.6 Rope Sling Load Pick Up 685
34.1.7 Rope Sling Load Drop Off 685
34.1.8 Fast Rope Insertion 686
34.1.9 Rope Suspension Extraction 686
34.1.10 Extract Using Rope 687
34.1.11 Rope Suspension Insertion 688
34.1.12 Ladder Extraction 688
34.1.13 Extract Using Ladder 690
34.1.14 Ladder Insertion 690
34.1.15 Insert Using Ladder 691
34.1.16 Dismount 691
34.2. Internal Transfer of Fuel 692
34.3. Transfer Supplies 692
34.4. IFF Tasks 692
34.4.1 IFF Interrogate 693
34.4.2 IFF Sweep Interrogate 694
34.5. Radio Tasks 695
34.5.1 Send Radio Set 695
34.5.2 Send Radio Task 696
34.5.3 Send Text Message 697

34.6. Wait Tasks ... 697
34.6.1 Wait ... 698
34.6.2 Wait Duration ... 698
34.6.3 Wait Elapsed ... 698
34.7. Tasks for Moving Articulated Parts ... 698
34.7.1 Open Car Door ... 698
34.7.2 Close Car Door ... 699
34.7.3 Open Cargo Door ... 699
34.7.4 Close Cargo Door ... 700
34.7.5 Deploy Refueling Boom ... 700
34.7.6 Stow Refueling Boom ... 700
34.7.7 Deploy Port or Starboard Refueling Drogue ... 701
34.7.8 Stow Port or Starboard Refueling Drogue ... 701
34.7.9 Extend Towed Array Decoy ... 702
34.7.10 Retract Towed Array Decoy ... 702
34.7.11 Raise Periscope ... 702
34.7.12 Lower Periscope ... 702
34.8. Embedded Entity Tasks ... 703
34.8.1 Deploying Embedded Entities ... 703
34.8.2 Recovering Deployed Entities ... 703
34.8.3 Sector Search Operation ... 703
34.8.4 Deploying an Entity by Assigning a Task ... 704
34.9. Deploy Sonobuoy ... 705
34.10. Deploy Sonobuoys Along Route ... 705
34.11. Sonar Dip ... 706
34.12. Release Next Entity From Queue ... 706
34.13. User Task ... 707
34.14. Reactive Tasks ... 707
34.15. Background Processes ... 709

34.1. Embarkation Tasks

This section describes tasks that simulate embarking and disembarking. Simulation objects move to the parent simulation object to embark, and move away after disembarking. For immediate embarkation and disembarkation, use the Embarked and Disembarked set data requests or the embarkation commands on the Objects menu. For a conceptual discussion of embarkation, see "15.1. Embarking and Disembarking Simulation Objects" on page 406.

34.1.1 Disembark

The Disembark task causes a simulation object to move to the parent simulation object's egress (exit) point and disembark. If the selected simulation object is a unit, the disembark task is given to all embarked subordinates. The task is considered complete when all subordinates are disembarked.

- When a simulation object disembarks, it goes to the disembarkation location. However, it does not move away from that location. If you plan to disembark several simulation objects from the same parent by giving them disembark tasks, you are responsible for making sure that each simulation object moves from the disembarkation point before the next simulation object disembarks.
- The Fast Rope Insertion task causes infantry to disembark from a helicopter. For details, see "34.1.8 Fast Rope Insertion" on page 686.

➤ **To assign a Disembark task:**

1. Select the simulation object you want to disembark.
2. Choose **Task > Embarkation > Disembark**, or click the Disembark button ().

34.1.2 Disembark All

The Disembark All task is given to a parent simulation object. It gives a Disembark task to all embarked simulation objects except aggregates. When you issue a Disembark All task, VR-Forces tries to disembark the simulation objects sequentially and have them move to sufficiently dispersed locations to avoid collision problems.

➤ **To assign a Disembark All task:**

1. Select the parent simulation object.
2. Choose **Task > Embarkation > Disembark All**.

34.1.3 Embark

The Embark task causes a simulation object to move to the ingress (entry) point of the parent simulation object and embark on it. If the embarking simulation object is a unit, each available taskable subordinate is given an embark task. When all subordinates have embarked, the task is considered complete.

The Embark task is valid only if the specified parent simulation object is configured to accept the simulation object that wants to embark on it. If you task a simulation object to embark on a simulation object that is not configured to receive it, for example, you task a DI to embark on an M1A2, the simulation object will not respond to the task. If you want to embark a simulation object on a simulation object that is not configured to receive it, you can use the immediate embarkation process. For details, see "15.1. Embarking and Disembarking Simulation Objects" on page 406.

Human characters have a set of animations that allow them to open and close car doors and get into and out of vehicles that have interior geometry and articulated doors. For details, see "Embarking Humans in Cars" on the facing page.

- If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later.
- If the embarkation spaces on the parent simulation object are full, VR-Forces sends a message to the sim engine console and the simulation object does not try to embark.
- If you give multiple simulation objects embark tasks for the same parent, you are responsible for spreading the tasks over time so that the simulation objects do not all converge on the parent at the same time, which would result in problems due to collision avoidance.
- A simulation object is considered embarked as soon as it reaches its ingress point. In plan view mode, its icon is hidden immediately. However, in a 3D mode, the simulation object may need to continue moving to reach its slot on the parent. If you task the parent simulation object to move before a simulation object has moved to its slot, the embarking 3D model may end up in an odd position relative to the parent simulation object. To avoid this visual problem, allow time for the embarking simulation object to stop moving before the parent entity moves.

➤ **To assign an Embark task:**

1. Select the simulation object.
2. Choose **Task > Embarkation > Embark**, or click the Embark button (). The Embark On dialog box opens.

   Unlike the Embark On dialog box that opens when you choose **Objects > Embarkation > Embark**, the Embark On dialog box for the Embark task does not allow you to choose the embarkation point.

3. Select the parent simulation object (the simulation object to embark on) or type its name in the Name box.
4. If the parent simulation object has slots of a specific type and you want the simulation object to embark in a slot of a particular type, select the type in the Slot Type list. Slots of that type are listed in the Slot Name list. (For information about slot types and slot names, see "73.3. Configuring Slots" on page 1444.)

5. Select a slot from the Slot Name list.
6. If you want the simulation object to use any available slot, select the Choose Any Slot if Requested Slot Not Available check box.
7. Click OK.



Embarking Humans in Cars

Some human characters have animations that allow them to open and close car doors and get into and out of cars. Humans get into and out of cars using the Embark and Disembark tasks.

When you give an entity an embark task for a car, you can specify that it embark in the driver's seat or a passenger seat. If you choose a passenger seat, you can specify which one to use. You do this by specifying the slot to embark in, as described in "34.1.3 Embark" on page 681.

If you do not specify the slot to use, the first entity that embarks in a car gets into the driver's seat. Additional entities embark in passenger seats until the number of slots available is used up.

The embark in cars capability is primarily intended for use with aerial surveillance scenarios. Viewing embarkation at ground level will not provide a completely realistic experience, including:

- You cannot change the speed at which entities enter and exit the vehicle and open and close the door. For example, they cannot run and jump into a car or jump out of a car and immediately run away.
- Contact between an entity's hand and the car door will not be entirely lifelike.
- If Show Occupants is enabled, an entity that embarks in the driver's seat will get in on top of the automatically displayed driver. We recommend you disable occupants for all cars or at least the car to be embarked on. For information about showing and hiding occupants, see "40.40. Hide Occupants" on page 882 and "51.4. Displaying DI-Guys in Vehicles" on page 1085.
- The entry and exit animations are calibrated for civilian vehicles. Trying to embark on a large vehicle, such as a truck that has doors that are high off the ground, is not taken into account.
- This release does not support drivers in the right front seat.

34.1.4 Airdrop

The Airdrop task is used by aircraft to disembark paratroopers or entities such as vehicles. It specifies only when the objects embarked on the aircraft will parachute, and it can specify the number of "drop slots" available to those entities.

On an aircraft, drop slots indicate the number of intervals at which entities can exit the aircraft with the Airdrop from Aircraft task. On entities, the drop slot indicates the order in which they will perform that task. Drop slots do not support units, and they are not the same as embarkation slots.

The aircraft performing the Airdrop task must have the Air Drop system (*airDrop.sysdef*). The embarked entities must have a task for deploying the parachute or they will not exit the aircraft. See "34.1.5 Airdrop from Aircraft" below for details.

> The movement task Parachute to Target does not coordinate with Airdrop.

➤ **To assign an airdrop task:**

1. Select the aircraft that the parachuting entities are embarked upon.
2. Choose **Task > Embarkation > Airdrop**. The Airdrop parameters display.
3. Indicate the number of seconds between drops. Entities will exit the aircraft at that interval.
4. Optionally, you can select the Use Drop Slots check box and then define the number of drop slots. One or more entities can exit the aircraft at each drop slot, as specified in the Airdrop from Aircraft task.
5. Click OK.

34.1.5 Airdrop from Aircraft

Paratrooper entities and simulation objects, such as ground vehicles, with the Air Drop Capable system (*airdrop-capable.sysdef*) can parachute from an aircraft on which they are embarked. The aircraft must have a task for airdropping the entity. For more information, see "34.1.4 Airdrop" on the previous page or "30.1. Airdrop Along Route" on page 585.

Drop slots are optional. If you use them, you should specify the drop slots in both the tasks for the entities on the aircraft (Airdrop from Aircraft, described below) and the aircraft itself (Airdrop or Airdrop Along Route). If the aircraft's airdrop task does not use drop slots, then the entities will parachute one at a time (regardless of any specified drop slots). If the aircraft's airdrop task has fewer drop slots defined than the drop slot specified for an entity, then that entity will not perform the airdrop. Units cannot use drop slots.

> While an entity is parachuting, if you double-click that entity to try to move it, it snaps to the terrain.

➤ **To assign an airdrop from aircraft task:**

1. Select the entity embarked on the aircraft.
2. Choose **Task > Embarkation > Airdrop from Aircraft**. The Airdrop from Aircraft parameters display.
3. Indicate whether the entity should deploy the parachute after a specific amount of time or at a particular altitude.
   - To specify that the entity should deploy the parachute seconds after exiting the aircraft, select the Deploy At Elapsed Time check box and then enter the number of seconds.

— If you choose to deploy at altitude, you can enter the altitude in meters. You can use the check box to indicate that the altitude be above ground level (default is sea level).

Be sure that the entity deploys the parachute at a point where it will open and slow their descent to the terrain.

4. If the entity is a vehicle or another simulation object with the Air Drop Capable system, you can specify the number of parachutes that the entity should use from the Cargo Parachute Type list. This setting does not affect the parachute for Paratrooper entities.
5. Optionally, specify a point where the entity should try to land.
6. Optionally, specify a drop slot. This is the order in which the entity will exit the aircraft if the Airdrop or Airdrop Along Route task includes drop slots. Multiple entities can use the same slot to exit the aircraft simultaneously.
7. Click OK.

34.1.6 Rope Sling Load Pick Up

The Rope Sling Load Pick Up task causes a helicopter to move to a specified location and then move to a designated cargo to load on the rope. Helicopters require the Fast Roping system.

➤ **To assign a Rope Sling Load Pick Up task:**

1. Select the helicopter.
2. Choose **Task > Embarkation > Rope Sling Load Pick Up**. The Rope Sling Load Pick Up parameters display.
3. Optionally, filter the Pickup Point list to view either Waypoints or Plan Variables.
4. Specify the location that the helicopter should move to.
5. Specify the cargo for the helicopter to pick up.
6. Click OK.

34.1.7 Rope Sling Load Drop Off

The Rope Sling Load Drop Off task causes a helicopter to move to a specified location and drop off cargo that it picked with the Rope Sling Load Pick Up task.

➤ **To assign a Rope Sling Load Drop Off task:**

1. Select the helicopter.
2. Choose **Task > Embarkation > Rope Sling Load Drop Off**. The Rope Sling Load Drop Off parameters display.
3. Specify the location where the helicopter should drop off the cargo.
4. Click OK.

34.1.8 Fast Rope Insertion

The Fast Rope Insertion task causes a helicopter to move to a specified waypoint and disembark a specified number of embarked entities using a fast rope animation. The disembarked entities slide down the rope, then move to a specified muster point and assume a kneeling posture.

This task is only available to helicopters that have the Fast Roping system, such as the MH-60 Black Hawk or the CH-47 Chinook.

➤ **To assign a Fast Rope Insertion task:**

1. Embark entities on a helicopter that supports the Fast Rope Insertion task.
2. Select the helicopter.
3. Choose **Task > Embarkation > Fast Rope Insertion**. The Fast Rope Insertion parameters display.
4. Optionally, filter the Insertion Point list to view either Waypoints or Plan Variables.
5. Specify the location at which the insertion should take place.
6. Specify the altitude at which the helicopter should hover.
7. Specify the number of entities to be disembarked and inserted.
8. Optionally, filter the Muster Point list to view either Waypoints or Plan Variables.
9. Specify the destination where the entities should move to after disembarking.
10. Click OK.

34.1.9 Rope Suspension Extraction

Helicopters with the Fast Roping system can perform a rope extraction. The Rope Extraction task specifies the waypoint where the helicopter will lower the rope and which entities to extract. Those entities must be within 50 meters of the rope and also have the Extract Using Rope task or they will not embark on the rope. For details, see "34.1.10 Extract Using Rope" on the facing page.

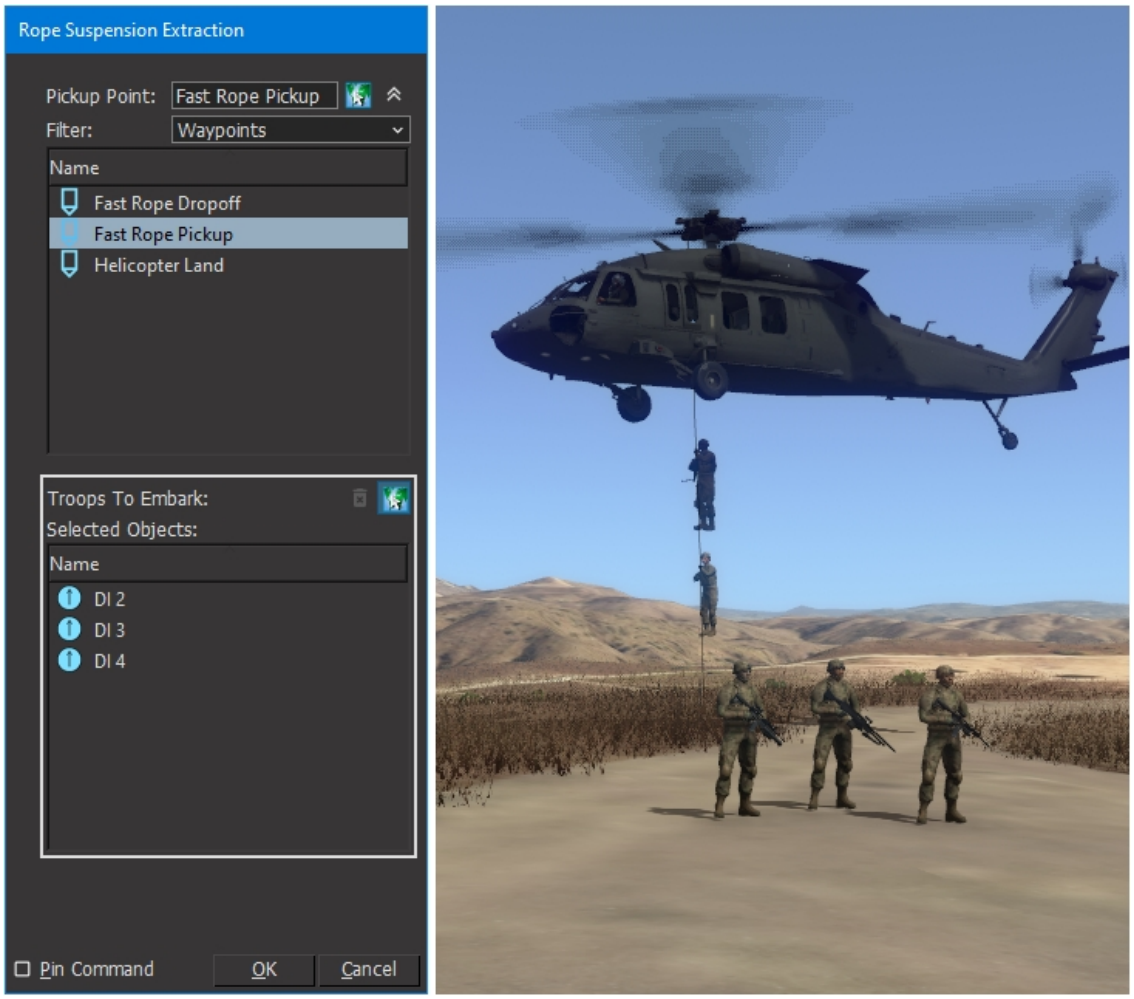
This task is only available to helicopters that have the Fast Roping system.

➤ **To have a helicopter extract entities with a rope:**

1. Select a helicopter that supports the Rope Extraction task.
2. Choose **Task > Embarkation > Rope Suspension Extraction**. The Rope Suspension Extraction parameters display (Figure 243).
3. Optionally, filter the Pickup Point list to view either Waypoints or Plan Variables.
4. In the Pickup Point list, specify the location from which the entities will be extracted.
5. Select the entities to extract in either the Objects List or in the window. They are added to the Troops To Embark list.
6. Click OK.

When the vehicle arrives at the waypoint and deploys the rope, entities on its Troops to Embark list that have the Extract Using Rope task and are within 50 meters of the rope will embark on the rope one at a time. After an entity attaches itself to the rope, the vehicle ascends slightly to allow the next entity to attach to the rope.

Figure 243. Rope Suspension Extraction



34.1.10 Extract Using Rope

Entities can be extracted via a rope deployed by a helicopter using the Rope Suspension Extraction task; see "34.1.9 Rope Suspension Extraction" on the previous page. In addition to the Extract Using Rope task, the entities must be included on the Troops to Embark list in the Rope Suspension Extraction task, and they must be within 50 meters of the point on the terrain where the rope was deployed.

Entities that have been extracted using the rope can only disembark from the rope via the Rope Suspension Insertion task. See "34.1.11 Rope Suspension Insertion" on the next page.

➤ **To specify that an entity will be extracted using a rope:**

1. Select the entity.
2. Choose **Task > Embarkation > Extract Using Rope**.

When the helicopter arrives and deploys the rope, surviving entities embark on the rope one at a time. After an entity attaches itself to the rope, the vehicle ascends slightly to allow the next entity to attach to the rope.

34.1.11 Rope Suspension Insertion

Helicopters with the Fast Roping system can perform an insertion via rope suspension using the Rope Suspension Insertion task. This applies to entities that are embarked on the helicopter's rope from an earlier extraction, and is the only way to disembark those entities. See "34.1.9 Rope Suspension Extraction" on page 686 to learn how to embark entities on the rope.

This task is different from the Fast Rope Insertion task, which is used for entities that begin while embarked inside a helicopter and then slide down a rope. For details, see "34.1.8 Fast Rope Insertion" on page 686.

This task is only available to helicopters that have the Fast Roping system.

➤ **To have a helicopter insert entities via rope suspension:**

1. Select a helicopter that supports the Rope Suspension Insertion task.
2. Choose **Task > Embarkation > Rope Suspension Insertion**. The Rope Suspension Insertion parameters display.
3. Optionally, filter the Dropoff Point list to see either Waypoints or Plan Variables.
4. In the Dropoff Point list, select the location where the entities will be inserted.
5. Click OK.

When the helicopter arrives at the waypoint, entities on the rope will disembark one at a time. After the entity at the bottom disembarks, the helicopter descends slightly so that the next entity on the rope can disembark.

34.1.12 Ladder Extraction

Helicopters with the Rope Ladder system can perform a ladder extraction. The Ladder Extraction task specifies the waypoint where it will drop the ladder and which entities to extract. Those entities must be within 50 meters of the ladder and also have the Extract Using Ladder task, or they will not climb the ladder and board the vehicle. For details, see "34.1.13 Extract Using Ladder" on page 690.

This task is only available to helicopters that have the Rope Ladder system, such as the CH-47 Chinook.

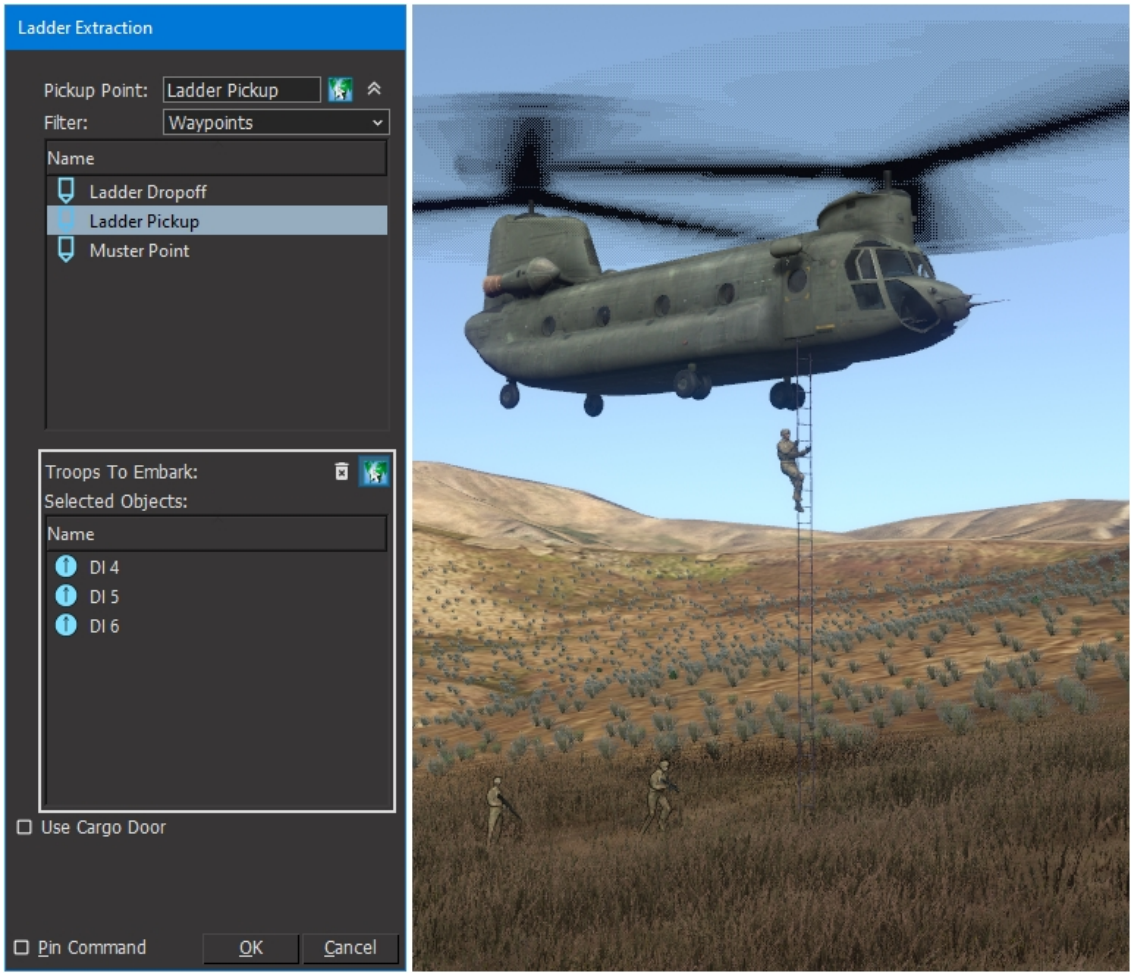
> This task has the option to open the vehicle's rear cargo door. You can use the Simulation Object Editor to edit the Rope Ladder system and attach it to the rear of the helicopter instead of the front crew door (Figure 245). In that case, the ladder would deploy from the back and then the entities ascend the ladder and embark on the helicopter using the cargo door. See "70.4.4 Editing a System's Properties" on page 1392 for details.

➤ **To have a helicopter extract entities with a ladder:**

1. Select a helicopter that supports the Ladder Extraction task.
2. Choose **Task > Embarkation > Ladder Extraction**. The Ladder Extraction parameters display (Figure 244).
3. Optionally, filter the Pickup Point list to view either Waypoints or Plan Variables.
4. In the Pickup Point list, specify the location from which the entities will be extracted.
5. Optionally, filter the Troops to Embark list to see either All Entities or Plan Variables.
6. Select the entities to extract in either the Objects List or in the window. They are added to the Troops To Embark list.
7. Optionally, select Use Cargo Door to open the vehicle's rear cargo door.
8. Click OK.

When the helicopter arrives at the destination and deploys the ladder, entities on its Troops to Embark list that have the Extract Using Ladder task and are within 50 meters of the ladder will climb it and embark on the vehicle.

Figure 244. Ladder Extraction



34.1.13 Extract Using Ladder

Entities can be extracted via a ladder deployed by a helicopter using the Ladder Extraction task. See "34.1.12 Ladder Extraction" on page 688. For an entity to successfully execute the Extract Using Ladder task, the entity must be included on the Troops to Embark list in the Ladder Extraction task and must be within 50 meters of the extraction point.

➤ **To specify that an entity will be extracted using a ladder:**

1. Select the entity.
2. Choose **Task > Embarkation > Extract Using Ladder**.

When the helicopter arrives at the destination and deploys the ladder, surviving entities climb it and embark on the vehicle.

34.1.14 Ladder Insertion

Helicopters with the Rope Ladder system can perform a ladder insertion. The Ladder Insertion task specifies the waypoint where it will drop the ladder. Entities embarked on the helicopter with the Insert Using Ladder task will descend the ladder and move into formation. For details, see "34.1.15 Insert Using Ladder" on the facing page. There is no restriction on how the entities were embarked on the helicopter.

This task is only available to helicopters that have the Rope Ladder system.

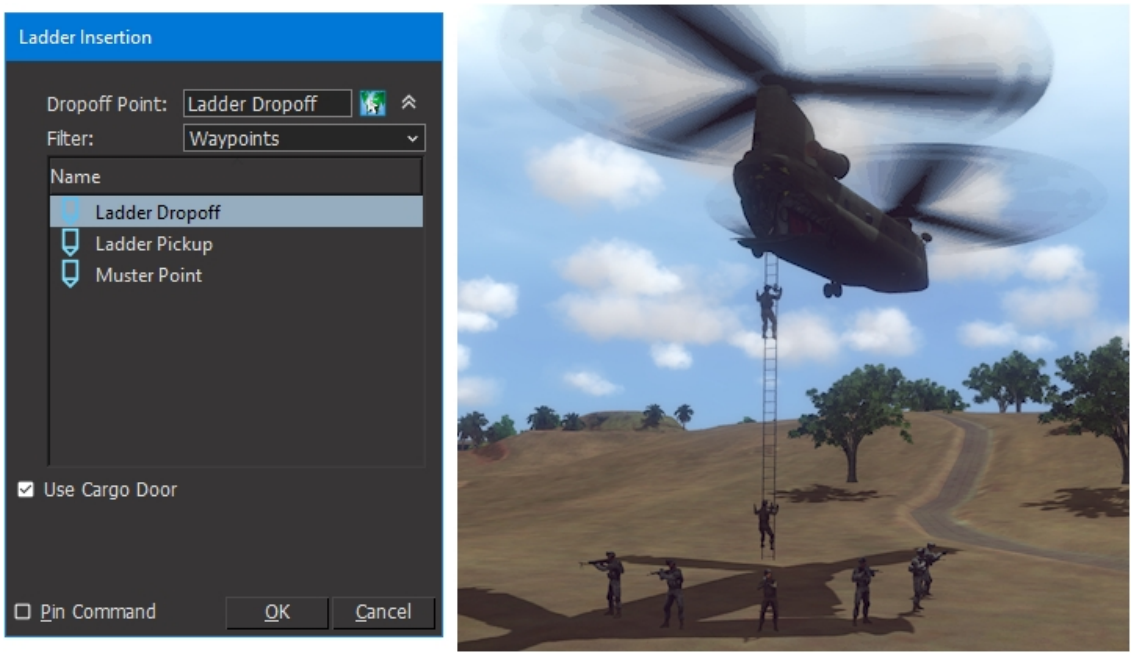
> This task has the option to open the vehicle's rear cargo door. You can use the Simulation Object Editor to edit the Rope Ladder system and attach it to the rear of the helicopter instead of the front crew door. In that case, the ladder deploys from the back and the entities descend the ladder from there (Figure 245). See "70.4.4 Editing a System's Properties" on page 1392 for details.

➤ **To have a helicopter insert entities with a ladder:**

1. Select a helicopter that supports the Ladder Insertion task.
2. Choose **Task > Embarkation > Ladder Insertion**. The Ladder Insertion parameters display (Figure 245).
3. Optionally, filter the Dropoff Point list to see either Waypoints or Plan Variables.
4. In the Dropoff Point list, specify the location where the entities will be inserted.
5. Optionally, select Use Cargo Door to open the vehicle's cargo door.
6. Click OK.

When the helicopter arrives at the destination and deploys the ladder, embarked entities that have the Insert Using Ladder task will climb down it and then move into formation.

Figure 245. Ladder Insertion with Use Cargo Door and with Rope Ladder system positioned at rear of vehicle



34.1.15 Insert Using Ladder

Entities can be inserted via a ladder deployed by a helicopter using the Ladder Insertion task. See "34.1.14 Ladder Insertion" on the previous page. The entities must be embarked on the helicopter and have the Insert Using Ladder task assigned to be included in the insertion. There is no restriction on how the entities were embarked on the helicopter.

➤ **To specify that an entity will be inserted using a ladder:**

1. Select an entity that is (or will be) embarked on the helicopter that has the Ladder Insertion task.
2. Choose **Task > Embarkation > Insert Using Ladder**.

When the helicopter arrives at the destination and deploys the ladder, each entity with this task will descend the ladder and then move into formation around its base.

34.1.16 Dismount

You can task squads or fire teams that are embarked on a rotary-wing aircraft or ground vehicle with the Dismount task. The entities will exit and assemble in formation around the aircraft or vehicle.

➤ **To task a squad or fire team to exit an aircraft or vehicle and assume formation:**

1. Select a squad or fire team unit.
2. Choose **Task > Embarkation > Dismount**.

34.2. Internal Transfer of Fuel

Entities with the Air Refueling Supplier system, typically tanker aircraft, can transfer their off-load fuel to their onboard fuel supply using the Internal Transfer of Fuel task. You can specify whether they should fill up to their full capacity or just a specific amount of fuel, as well as the rate at which the fuel supply should be replenished.

➤ **To task a tanker aircraft to transfer fuel to itself:**

1. Select the tanker aircraft.
2. Choose **Task > Logistics > Internal Transfer of Fuel**. The Internal Transfer of Fuel parameters display.
3. Specify the parameters for the transfer.
4. Click OK.

34.3. Transfer Supplies

Entities with an air refueling receiver systems can use the Transfer Supplies task to transfer fuel supplies from another entity to itself. The task uses the DIS and HLA RPR FOM Logistics Resupply protocol. The specified supplier must have that fuel resource for the task to be successful.

➤ **To assign the Transfer Supplies task:**

1. Select the entity that will receive the fuel supply.
2. Choose **Task > Logistics > Transfer Supplies**. The Transfer Supplies parameters display.
3. From the Supply Type list, select the fuel type.
4. If you want the entity to fill its fuel resource, select the Fill Up to Capacity check box.

   If you do not want the entity to fill its fuel resource to capacity, clear the check box and then enter the amount it should take in the Supply Amount box.
5. Enter the rate at which the supply should transfer.
6. From the Supplier list, select the entity that will supply the fuel.
7. Click OK.

34.4. IFF Tasks

You can task entities configured with an IFF interrogator system can to interrogate other entities' IFF status. VR-Forces uses open standards such as RPR FOM and DIS to query the IFF transponder status of the other entities.

VR-Forces includes the ability to perform a Lethal Mode 5 interrogation on entities to check for Mode 5 enabled transmitters. It also displays a **V** next to any entities that are the target of a Lethal Mode 5 interrogation when they have not enabled Mode 5.

By default, there is no simulation effect in the scenario. The IFF information sent is the VR-Link interrogation response, which is simply the IFF modes that are enabled on the target. That information is published to the network, making it available for higher fidelity IFF environments, such as an AWAC simulator. In the GUI, you can see entity's IFF data in the Information dialog box and Last Selected Entity panel.

- Set entities' IFF codes using the IFF set data request as described in "40.41. IFF" on page 883.
- Clutter aircraft include IFF information. See "16.3.1 Aircraft Clutter Area" on page 428.

34.4.1 IFF Interrogate

You can task an aircraft entity to interrogate another entity for IFF transponders.

Any enabled modes display in the interrogating entity's Object Console as warnings (Figure 246). When you enable lethal interrogation for Mode 5, if the target of the interrogation has Mode 5 available but does not have it enabled, the interrogated entity briefly displays a **V** in the Plan View and the Object Console indicates it was a lethal interrogation (Figure 247).

Figure 246. IFF interrogation results

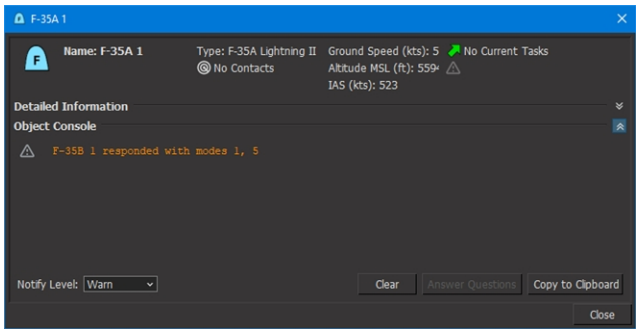
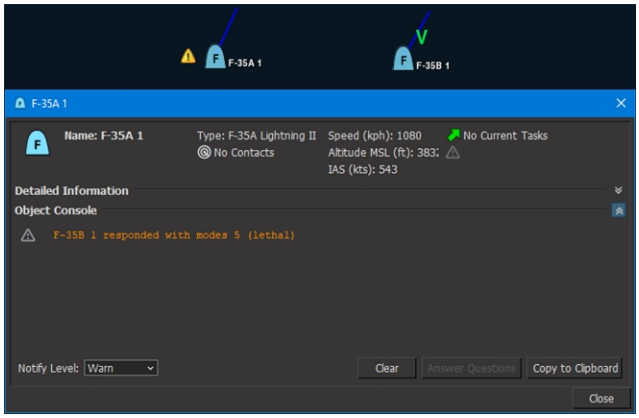


Figure 247. IFF lethal interrogation results



VR-Forces implements the NETN FOM, based on RPR FOM 2.0, with IFF extensions from RPR FOM 3.0. If you use HLA Evolved, use the FOM provided by MAK.

There is no real simulation effect in the scenario. The resulting information is only published to the network, making it available for higher fidelity IFF environments.

➤ **To assign an IFF Interrogate task:**

1. Select the entity to task.
2. Choose **Task > IFF > IFF Interrogate**. The IFF Interrogate parameters display.
3. Use Name box to specify the entity to interrogate.
4. Select the modes that you want to check for.
5. Click OK.

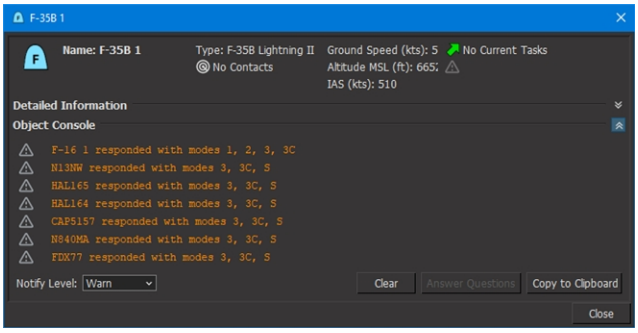
34.4.2 IFF Sweep Interrogate

You can task an aircraft entity to interrogate entities within range for their IFF transponders. It returns the modes that are present and on. Detected modes display in the interrogating entity's Object Console as warnings.

In the task, you specify the area that the entity should sweep. This is a combination of the range and the sweep. There are three methods for defining the sweep:

- 360 Degree Sweep. Interrogates all aircraft entities within the specified range.
- Sweep Range. Specify an offset from the orientation.
- From Boresight. Specify an angle from the entity's boresight.

Figure 248. IFF sweep interrogation results



VR-Forces implements the NETN FOM, based on RPR FOM 2.0, with IFF extensions from RPR FOM 3.0. If you use HLA Evolved, use the FOM provided by MAK.

There is no real simulation effect in the scenario. The resulting information is only published to the network, making it available for higher fidelity IFF environments.

➤ **To assign an IFF Sweep Interrogate task:**

1. Select the entity to task.
2. Choose **Task > IFF > IFF Sweep Interrogate**. The IFF Sweep Interrogate parameters display.
3. Enter the parameters for the sweep.
4. Select the modes that you want to check for.
5. Click OK.

34.5. Radio Tasks

This section describes the tasks that allow you to use the VR-Forces radio network to send messages, tasks, and set data requests.

34.5.1 Send Radio Set

The Send Radio Set task lets you send a set data request to a simulation object using a VR-Forces radio message. The receiving entity executes the set data request as if you had issued it directly to the entity from the Set menu or a set data request in a plan.

ⓘ Since VR-Forces has no idea what type of entity you might want to give a set data request to, when you add this task the full list of possible sets is available. There is no context sensitivity like there is when you select a simulation object and view the Set menu. It is up to you to be sure that the set you are assigning to a simulation object is supported by that object type.

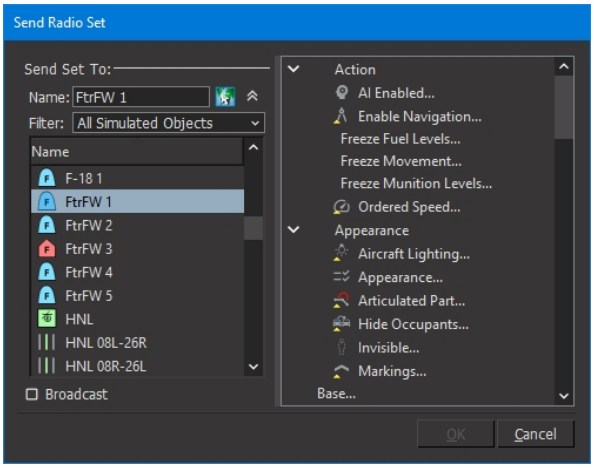
➤ **To assign a Send Radio Set task:**

1. Select the simulation object that will send the message.
2. Choose **Task > Radio > Send Radio Set**. The Send Radio Set parameters display.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, see "80.2. Configuring Radios" on page 1574.)

   To send the message to a specific simulation object, in the Send Set To group box, in the Name text box, type the name of the simulation object that will receive the set data request message, or select the simulation object in the list or on the terrain.

   There is a list of set data requests to choose from (Figure 249).

Figure 249. Send Radio Set dialog box



4. Select the set data request you want to send.
5. Click OK. If the set data request requires input, an appropriate dialog box opens. If no input is required, the task setup is finished.
6. If a dialog box opens, complete the dialog box as described in chapters 34 and 35.
7. Click OK.

34.5.2 Send Radio Task

The Send Radio Task task lets you send a task to a simulation object using a VR-Forces radio message. The receiving simulation object executes the task as an independent task.

> Since VR-Forces has no idea what type of simulation object you might want to give a task to, when you add this task the full list of possible tasks is available. There is no context sensitivity like there is when you select a simulation object and view the Task menu. It is up to you to be sure that the task you are assigning to a simulation object is supported by that object type.

➤ **To assign a Send Radio Task:**

1. Select the simulation object that will send the message.
2. Choose **Task > Radio > Send Radio Task**. The Send Radio Task parameters display.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, see "80.2. Configuring Radios" on page 1574.)

   To send the message to a specific simulation object, in the Send Task To group box, in the Name text box, type the name of the simulation object that will receive the task message, or select the simulation object in the list or on the terrain.

   A list of tasks displays (similar to Figure 249 above).
4. Select the task you want to send.

5. Click OK. If the task requires input, an appropriate dialog box opens. If no input is required, the task setup is finished.
6. If a dialog box opens, complete the dialog box as described in the various task procedure chapters.
7. Click OK.

34.5.3 Send Text Message

The Send Text Message task lets you send a text message to a simulation object using a VR-Forces simulated radio message. The receiving simulation object must be on the same radio network as the sending simulation object. Usually this means the same unit or same force. The receiving simulation object can test for receipt of the message in a conditional statement in its plan. (For details, see "42.2.6 Conditional Tests" on page 939.)

> If you want to send a text message that can be received by simulation objects in a different radio network from the sender, use the Send Text Message command. (For details, see "43.7.2 Sending Text Messages from the Commands Menu" on page 972.)

➤ **To assign a Send Text Message task:**

1. Select the simulation object that will send the message.
2. Choose **Task > Radio > Send Text Message**. The Send Text parameters display.
3. To send the message to all simulation objects configured to receive messages from this simulation object's radio, select the Broadcast check box. (For details about configuring the radio model, see "80.2. Configuring Radios" on page 1574.)

   To send the message to a specific simulation object, in the Name text box, type the name of the simulation object that will receive the message, or select the simulation object in the list or on the terrain.
4. In the Message Text box, type the message.
5. Click OK.

34.6. Wait Tasks

You can order a simulation object to wait indefinitely, for a specified period of time, or until a specified elapsed simulation time. You can also order a human to wait in a queue. For details, see "32.32. Wait in Queue" on page 664.

> The Wait task is not equivalent to an immediate stop task. If a simulation object is moving and you give it a Wait task, it might not stop immediately, particularly if a controller such as the collision avoidance controller is controlling its movement.

34.6.1 Wait

➤ **To assign a Wait task:**

1. Select the simulation object.
2. Choose **Task > Wait > Wait**.

If you put an indefinite wait in a plan, the simulation object does not progress through its plan until it receives another command.

34.6.2 Wait Duration

➤ **To assign a Wait Duration task:**

1. Select the simulation object.
2. Choose **Task > Wait > Wait Duration**. The Wait Duration parameters display.
3. Specify the duration of the wait in hours, minutes, and seconds.
4. Click OK.

34.6.3 Wait Elapsed

➤ **To assign a Wait Elapsed task:**

1. Select the simulation object.
2. Choose **Task > Wait > Wait Elapsed**. The Wait Elapsed parameters display.
3. Specify the elapsed time from the start of the simulation, in hours, minutes, and seconds.
4. Click OK.

For information about simulation time and simulated exercise time, see "3.4.1 Simulation Time" on page 120.

34.7. Tasks for Moving Articulated Parts

Some entity models have articulated parts, such as doors and periscopes, that you can move.

34.7.1 Open Car Door

Vehicles that have a Car Door system and whose model includes articulated parts for the car doors can open and close their doors. Open Car Door opens a specific door on a vehicle.

➤ **To open a car door:**

1. Select a vehicle that has a Car Door system.
2. Choose **Task > Car Door > Open Car Door**. The Open Car Door dialog box opens.

3. In the Door list, select the door that you want to open.
4. Optionally, in the Swing Rate box, specify the speed at which you want the door to open.
5. Click OK.

34.7.2 Close Car Door

Vehicles that have a Car Door system and whose model includes articulated parts for the car doors can open and close their doors. Close Car Door closes a specific door on a vehicle.

➤ **To close a car door:**

1. Select a vehicle that has an open door.
2. Choose **Task > Car Door > Close Car Door**. The Close Car Door dialog box opens.
3. In the Door list, select the door that you want to close.
4. Optionally, in the Swing Rate box, specify the speed at which you want the door to close.
5. Click OK.

34.7.3 Open Cargo Door

The Open Cargo Door task opens a cargo door on an entity that has a cargo door articulated part.

Figure 250. CH-53E Super Stallion with open cargo door



➤ **To assign an Open Cargo Door task:**

1. Select the entity.
2. Choose **Task > Cargo Door > Open Cargo Door**.

34.7.4 Close Cargo Door

The Close Cargo Door task closes a cargo door on an entity that has a cargo door articulated part.

➤ **To assign a Close Cargo Door task:**

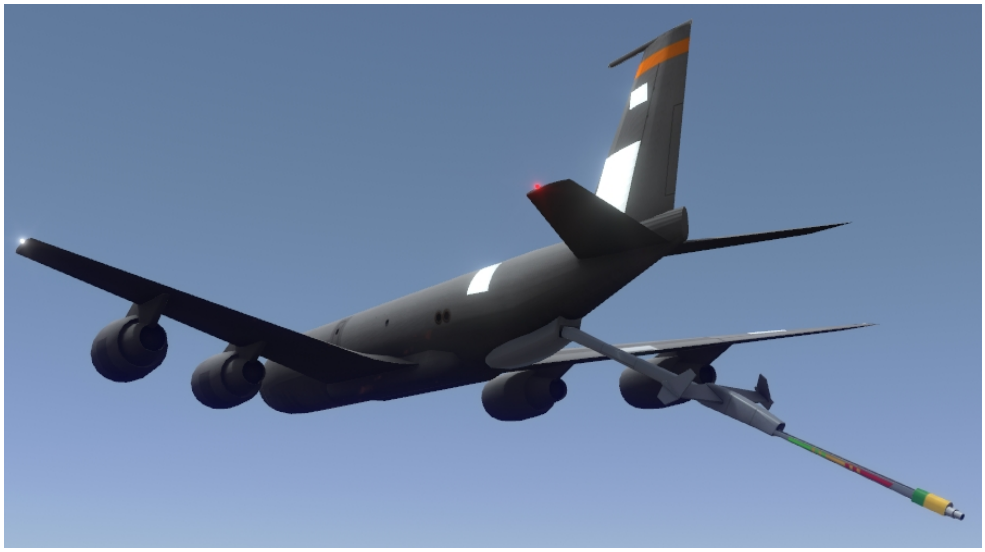
1. Select the entity.
2. Choose **Task > Cargo Door > Close Cargo Door**.

34.7.5 Deploy Refueling Boom

The Deploy Refueling Boom task causes an aircraft that has a refueling boom articulated part to extend it (Figure 251). Use Stow Refueling Boom (see "34.7.6 Stow Refueling Boom" below) to retract the boom.

- This task only deploys the boom. See "30.7. Establish Refueling Station" on page 587 for the refueling task.
- To use the task, the entity must have the appropriate refueling system. To visualize the task, the model must be configured with the articulated part.

Figure 251. KC-135 Stratotanker with refueling boom deployed



➤ **To assign a Deploy Refueling Boom task:**

1. Select the entity.
2. Choose **Task > Refueling Parts > Deploy Refueling Boom**. The boom extends.

34.7.6 Stow Refueling Boom

The Stow Refueling Boom task causes an aircraft that has a refueling boom articulated part to stow it. Use Deploy Refueling Boom (see "34.7.5 Deploy Refueling Boom" above) to deploy a refueling boom.

➤ **To assign a Stow Refueling Boom task:**

1. Select the entity.
2. Choose **Task > Refueling Parts > Stow Refueling Boom**. The boom contracts.

34.7.7 Deploy Port or Starboard Refueling Drogue

The Deploy Port Refueling Drogue and Deploy Starboard Refueling Drogue tasks cause an aircraft that has a port-or starboard-side refueling drogue articulated part to extend the specified drogue (Figure 252). Use Stow Port Refueling Drogue or Stow Starboard Refueling Drogue (see "34.7.8 Stow Port or Starboard Refueling Drogue" below) to retract the drogue.

- These tasks only extend a drogue. See "30.7. Establish Refueling Station" on page 587 for the refueling task.
- To use the task, the entity must have the appropriate refueling system. To visualize the task, the model must be configured with those articulated parts.

Figure 252. KC-130 Hercules Tanker with port and starboard drogues deployed



➤ **To assign a deploy refueling drogue task:**

1. Select the entity.
2. Choose **Task > Refueling Parts > Deploy Port Refueling Drogue** or **Deploy Starboard Refueling Drogue**. The drogue extends.

34.7.8 Stow Port or Starboard Refueling Drogue

The Stow Port Refueling Drogue and Stow Starboard Refueling Drogue tasks cause an aircraft that has a port-or starboard-side refueling drogue articulated part to stow it. Use Deploy Port Refueling Drogue and Deploy Starboard Refueling Drogue (see "34.7.7 Deploy Port or Starboard Refueling Drogue" above) to deploy a refueling drogue.

➤ **To assign a stow refueling drogue task:**

1. Select the entity.
2. Choose **Task > Refueling Parts > Stow Port Refueling Drogue** or **Stow Starboard Refueling Drogue**. The drogue retracts.

34.7.9 Extend Towed Array Decoy

The Extend Towed Array Decoy task causes an aircraft that is configured with a towed array decoy articulated part to deploy it. Use Retract Towed Array Decoy (see "34.7.10 Retract Towed Array Decoy" below) to retract the decoy.

ⓘ To use the task, the entity must have the Towed Array Decoy system. To visualize the task, the model must be configured with the articulated part.

➤ **To assign an extend towed array decoy task:**

1. Select the entity.
2. Choose **Task > Other > Extend Towed Array Decoy**. The array extends.

34.7.10 Retract Towed Array Decoy

The Retract Towed Array Decoy task causes an aircraft that is configured with a towed array decoy articulated part to retract it. Use Extend Towed Array Decoy (see "34.7.9 Extend Towed Array Decoy" above) to deploy the decoy.

➤ **To assign a retract towed array decoy task:**

1. Select the entity.
2. Choose **Task > Other > Retract Towed Array Decoy**. The array extends.

34.7.11 Raise Periscope

The Raise Periscope task raises the periscope of a subsurface entity. The periscope is an articulated part that is visible in the Stealth observer mode.

➤ **To assign a Raise Periscope task:**

1. Select the entity.
2. Choose **Task > Other > Raise Periscope**.

34.7.12 Lower Periscope

The Lower Periscope task lowers the periscope of a subsurface entity. The periscope is an articulated part that is visible in the Stealth view.

➤ **To assign a Lower Periscope task:**

1. Select the entity.
2. Choose **Task > Other > Lower Periscope**.

34.8. Embedded Entity Tasks

You can deploy and recover embedded entities. You can also task an embedded entity to conduct a sector search. For details about embedded entities, see "15.2. Embedded Entities" on page 411.

34.8.1 Deploying Embedded Entities

To add an embedded entity to a simulation, you deploy it from the parent entity. You can deploy an embedded entity explicitly or implicitly (by assigning a task to it). An embedded entity's label displays its name and the name of its parent.

> There is a delay of a few seconds from the time a deployment is ordered and the time it is executed.

➤ **To assign a Deploy task:**

1. Select the parent entity.
2. Choose **Task > Embedded Entity > Deploy**. The Deploy parameters display.
3. Select the entity you want to deploy.
4. Click OK. The embedded entity deploys next to the parent entity in the relative position in its configuration.

34.8.2 Recovering Deployed Entities

If you want an embedded entity that has been deployed to return to its parent, you can have the parent recover it. If you do not recover it, the deployed entity continues to function in the scenario just like any other entity.

➤ **To assign a Recover task:**

1. Select the parent entity.
2. Choose **Task > Embedded Entity > Recover**. The Recover Embedded Entity parameters display.
3. Select the entity you want to recover.
4. Click OK. The parent sends a message to the deployed entity to return. It returns and is embedded again.

34.8.3 Sector Search Operation

The Sector Search Operation task causes an embedded entity to fly a sector search and rescue pattern. In addition to the basic search and rescue parameters, you have these options:

- If there are no undeployed embedded entities available, divert an already deployed entity to do the search.

- Force the parent entity to wait until the search is complete and the embedded entity is recovered until it continues with its plan or allow the parent entity to continue with its plan while the search is underway.

➤ **To assign a Sector Search Operation task:**

1. Choose an entity that has embedded entities.
2. Choose **Task > Sector Search Operation**. The Sector Search Operation parameters display.
3. Select the Allow Diversion check box if you want a deployed entity to be diverted to this task if necessary.
4. Select the Recover When Finished check box if you want the parent entity to wait until it recovers the deployed entity before it continues with its plan.
5. Specify the starting point of the search.
6. In the Leg box, specify the length of each leg of the search.
7. In the Altitude box, specify the altitude the search aircraft should fly at.
8. In the Air Speed box, specify the speed of the search aircraft.
9. Click OK.

34.8.4 Deploying an Entity by Assigning a Task

You can assign a task to a deployable entity, that is, an embedded entity that has not been deployed yet. This implicitly deploys the entity so that it can execute the task. You can assign a task to an embedded entity in the parent's plan or you can directly assign a task to the embedded entity. The Sector Search example scripted task is an example of how to task a parent entity to deploy an embedded entity.

ⓘ There is a delay of a few seconds from the time a deployment is ordered and the time it is executed.

➤ **To assign a task to a deployable entity in a parent's plan:**

1. Select the parent entity.
2. Choose **Objects > Plan**. The Plan window opens.
3. Choose **Embedded Entities >** *entity* **> Task** (or **Set**) **>** *task* (or *set*), where *entity* is one of the entities that the parent can deploy and *task* (or *set*) is one of the tasks or sets on the Task or Set menu. The parameters for the selected task or set display.

   ⓘ If an entity does not have embedded entities, the Embedded Entities menu is not available.

4. Fill out the dialog box like you would for any task or set.
5. Click OK. The task is added to the plan. When you run the scenario, the embedded entity deploys and is assigned the task or set.

When you deploy an entity from the Objects menu, you can continue to set its state or assign tasks using this method or you can select the entity itself and issue tasks and sets from the Task and Set menus. The logic of assigning tasks from the Objects menu's Embedded Entities submenu rather than directly, is that the parent entity issues the tasks to the deployed entity.

➤ **To deploy an embedded entity by assigning it a task:**

1. Select the parent entity.
2. Choose **Objects > Embedded Entities >** *entity* **> Task** (or **Set**) **>** *task* (or *set*), where *entity* is one of the entities that the parent can deploy and *task* (or *set*) is one of the tasks or sets on the Task or Set menu. An appropriate dialog box for the task or set opens.
3. Fill out the dialog box like you would for any task or set.
4. The entity is deployed. If you assigned it a task, it begins to execute its task. If you set its state, it is deployed with the updated state setting.

34.9. Deploy Sonobuoy

The Deploy Sonobuoy task enables an aircraft to deploy a sonobuoy for taking sonar readings. A sonobuoy is deployed as an entity. This task is available to entities that have a Sonobuoy Deployer system configured.

By default, the deployed sonobuoys are set to send spot reports to the entity that deployed them. You can specify that they send the spot reports to other entities, as well. They are always sent to the deploying entity.

ⓘ RPR FOM 1.0 does not support active sonar. You will not be able to create sonobuoys if they have an active sonar system.

➤ **To assign a Deploy Sonobuoy task:**

1. Select the entity.
2. Choose **Task > Deploy Sonobuoy**. The Deploy Sonobuoy parameters display.
3. Select the type of sonar to use – Active or Passive.
4. Specify the depth at which to deploy the sonobuoy.
5. Select the entities to send spot reports to. If you do not select any entities, the spot reports are sent only to the deploying entity.
6. Click OK.

34.10. Deploy Sonobuoys Along Route

The Deploy Sonobuoys Along Route task enables an aircraft to deploy sonobuoys as it moves along a route. A sonobuoy is deployed as an entity. This task is available to entities that have a Sonobuoy Deployer system configured.

By default, the deployed sonobuoys are set to send spot reports to the entity that deployed them. You can specify that they send the spot reports to other entities, as well. They are always sent to the deploying entity.

> RPR FOM 1.0 does not support active sonar. You will not be able to create sonobuoys if they have an active sonar system.

➤ **To assign a Deploy Sonobuoys Along Route task:**

1. Select the entity.
2. Choose **Task > Deploy Sonobuoys Along Route**. The Deploy Sonobuoys Along Route parameters display.
3. Select the route to use.
4. Specify the distance between each sonobuoy.
5. Select the type of sonar to use – Active or Passive.
6. Specify the depth at which to deploy the sonobuoys.
7. Select the entities to send spot reports to. If you do not select any entities, the spot reports are sent only to the deploying entity.
8. Click OK.

34.11. Sonar Dip

Rotary-wing entities that have sonar can dip the sonar into water at a specified depth. Dipped sonar is not visualized in the Stealth view. The sonar is just simulated at the specified depth. Entities cannot dip sonar if they are moving faster than a set speed.

➤ **To assign a Sonar Dip task:**

1. Select the entity.
2. Choose **Task > Sonar Dip**. The Sonar Dip parameters display.
3. Specify the location at which to dip the sonar.
4. Specify how long the sonar should stay dipped.
5. Select the type of sonar to dip – Active or Passive.

   > RPR FOM 1.0 does not support active sonar.

6. Specify the depth for the sonar.
7. Click OK.

34.12. Release Next Entity From Queue

The Release Next Entity From Queue task is sent to a queue tactical graphic and releases the entity at the head of the queue to move to the next task in its plan.

You can assign this task directly, as described in this procedure, or from the **Commands > Send Task** command in a plan. If you are trying to create a scenario that involves crowd queueing behaviors, then using a plan to control the queue is the more likely approach.

When an entity is released from the queue, the Wait in Queue task ends. If the entity does not have another task in its plan, it remains in its location. The next entity in the line moves to the head of the queue. This might result in some visual anomalies.

➤ **To assign a Release Next Entity From Queue task:**

1. Select the queue tactical graphic.
2. Choose **Task > Release Next Entity From Queue**.

34.13. User Task

User tasks are tasks that have been added to VR-Forces using the VR-Forces toolkit. To assign a user task, you must know the name of the task and the required values for up to four parameters.

The User Task command is not available on the Task menu unless a developer adds a controller that accepts a user task.

➤ **To assign a user task:**

1. Select the simulation object that you want to execute the task.
2. Choose **Task > Other > User Task**. The User Task parameters display.
3. Type the name of the task.
4. Type any required parameters.
5. Click OK.

34.14. Reactive Tasks

Reactive tasks are not listed individually because most of them run automatically; typically, no user action is required to execute them.

You can disable or enable reactive tasks as needed; see "29.5. Reactive Tasks" on page 572 for instructions.

Table 32 lists some of the reactive tasks provided with VR-Forces.

Table 32. Reactive tasks included with VR-Forces (partial list)

Reactive Task	Description
Automatic Headlights	Turn headlights and taillights on and off based on local lighting conditions. This is disabled by default. You must enable this reactive task to have he lights come on automatically.
Automatic Jamming	Causes an entity to automatically respond to being tracked by radar.
Automatic Look	Causes humans to randomly look at other humans.
Auto-Enable Show Terrain	Turns on Show Terrain task for object upon creation.
Autonomous Fire reactive tasks	Looks for valid targets for any of the weapon systems that are enabled to fire autonomously. Prioritizes those targets and sends a fire task for the top priority one.
Avoid Other Ships	Considers ship type: Aircraft carrier is higher than fishing boat (assumes nets are out), which is higher than sailboat, which is higher than other. Will avoid surfaced submarines.
Bingo Fuel Monitor	Monitors fuel on embedded entities and requests a recovery when bingo fuel level is reached.
Evade Missile	Causes fixed-wing entities to try to evade missiles.
Flee From threat reactive tasks	Entity will run from explosions, the source of gun fire, entities killed nearby, and objects that match the configured entity types. Flee from Fire Events is disabled by default, but you can enable it for entities in the Simulation Object Editor (SOE).
IADS Launcher Control	Automates the launcher of an Integrated Air Defense system. Communicates with a control station running the IADS Command Station script.
IFF Code Changer	Automatically changes the IFF code every half hour based on the time of day.
Make Way for Emergency	If an emergency vehicle that has its lights on approaches a ground vehicle that is driving on a road, the vehicle pulls over to the side of the road.
Suppressed Crouch	Forces entity to crouch when suppressed. Returns to previous posture when no longer suppressed. Does not otherwise interrupt movement.
Suppressed Prone	Forces the entity to stop and fall prone when suppressed to level 2 or 3. Resumes movement at old posture when no longer suppressed.
Visit Interest Points	Causes civilians to visit nearby civilian visit points.

34.15. Background Processes

EntityLevel.sms has these background processes:

- Delete Created Pedestrians. Deletes pedestrians when a pedestrian area is deleted.
- Flare Fade. Gradually fades out illumination flares.
- Queue Manager. Manages queuing for the Wait in Queue task.



35. Tasks for Aggregate-Level Scenarios

This chapter describes tasks that apply to aggregate-level scenarios.

35.1. Introduction to Aggregate-Level Tasks 713
35.2. Engagement Tasks 713
35.2.1 Air-to-Air Attack 713
35.2.2 Air-to-Ground Air Base Attack 713
35.2.3 Air-to-Ground Attack 714
35.2.4 Attack by Fire 714
35.2.5 Attack with Anti-Air Missile 715
35.2.6 Attack with Anti-Ship Missile 715
35.2.7 Attack with Land-Attack Missile 715
35.2.8 Attack with Torpedo 716
35.2.9 Automatic Air Defense 716
35.2.10 Biological Attack 716
35.2.11 Bomb Location 717
35.2.12 Chemical Attack 717
35.2.13 Indirect Fire 718
35.2.14 Nuclear Attack 718
35.3. Combat Engineering Tasks 719
35.3.1 Breach Obstacles 719
35.3.2 Construct Abatis 720
35.3.3 Construct Anti-Tank Ditch 721
35.3.4 Construct Barbed Wire 721
35.3.5 Construct Barricade 722
35.3.6 Construct Berm 722
35.3.7 Construct Booby Trap 722
35.3.8 Construct Bridge 723
35.3.9 Construct Dry Gap 723
35.3.10 Construct Fortified Area 724

35.3.11 Construct Fortified Line ... 724
35.3.12 Construct Minefield ... 725
35.3.13 Construct Strong Point ... 725
35.3.14 Destroy Bridge ... 726
35.3.15 Destroy Ditch ... 726
35.3.16 Destroy Explosive ... 727
35.3.17 Destroy Fortification ... 727
35.3.18 Destroy Obstacle ... 727
35.3.19 Improve Booby Trap ... 727
35.3.20 Improve Breach ... 728
35.3.21 Improve Bridge ... 728
35.3.22 Improve Ditch ... 728
35.3.23 Improve Fortification ... 729
35.3.24 Improve Obstacle ... 729
35.4. Air Base Tasks ... 730
35.4.1 Launch Flight ... 730
35.4.2 Launch QRA Aircraft ... 731
35.4.3 Prepare QRA ... 731
35.5. Movement Tasks for Aggregate-Level Scenarios ... 732
35.5.1 Change MOPP Level ... 732
35.5.2 Change Posture ... 733
35.5.3 Escort Air Unit ... 733
35.5.4 Halt Movement ... 733
35.5.5 Land at Air Base ... 734
35.5.6 Move ... 734
35.5.7 Move Along Route ... 734
35.5.8 Move Along Route Retrograde ... 734
35.5.9 Move To ... 734
35.5.10 Move to Retrograde ... 735
35.5.11 Move to Location (Plan Along Roads) ... 735
35.5.12 Move to Waypoint (Plan Along Roads) ... 735
35.5.13 Patrol Route ... 736
35.5.14 Patrol Between ... 736
35.5.15 Turn To Heading ... 736
35.5.16 Wait for Escort ... 736
35.5.17 Wait Tasks ... 736
35.6. Embarkation Tasks ... 737
35.6.1 Disembark ... 737
35.6.2 Disembark All ... 737
35.6.3 Embark ... 737
35.7. Radio Tasks ... 737

35.7.1 Send Radio Set 737
35.7.2 Send Radio Task 737
35.7.3 Send Text Message 737
35.8. Other Tasks 737
35.8.1 Activate Jammer 737
35.8.2 Deactivate Jammer 737
35.8.3 Drop Fuel Tanks 738
35.8.4 FASCAM 738
35.8.5 Refuel from Tanker 738
35.8.6 Refueling Mission 739
35.8.7 Send NBC Report 739
35.8.8 Send Obstacle Report 739
35.8.9 User Task 740
35.9. Background Process List 740

35.1. Introduction to Aggregate-Level Tasks

This chapter describes tasks that simulation objects in aggregate-level scenarios can run. Some tasks are identical to tasks supported by simulation objects in entity-level scenarios.

35.2. Engagement Tasks

The following topics describe tasks in the engagement category that apply to aggregate-level scenarios.

35.2.1 Air-to-Air Attack

The Air-to-Air Attack task instructs an air unit to attack another air unit.

➤ **To assign an air-to-air attack:**

1. Select an air unit.
2. Choose **Task > Engagement > Air-to-Air Attack**, or click the Air-to-Air Attack button (). The Air-to-Air Attack parameters display.
3. Select the target unit.
4. Optionally, in the Maximum Engagement Distance box, specify the maximum distance from which the unit can attack the target. If it is set to 0 (zero), the maximum range is the maximum range of the weapon.
5. Click OK.

35.2.2 Air-to-Ground Air Base Attack

The Air-to-Ground Air Base Attack task causes a unit to move along an ingress route, move to a target point, make HE attack(s) against an air base, and then move along the

egress route. The unit attacks with each different HE ammunition type in succession. For each ammunition type, the unit expends all of its ammunition. There is a 30-second delay between attacks with different ammunition types as the unit prepares to attack again.

➤ **To order an air-to-ground air base attack:**

1. Select an aircraft unit.
2. Choose **Task > Engagement > Air-to-Ground Air Base Attack**, or click the Air-to-Ground Air Base Attack button (). The Air-to-Ground Air Base Attack parameters display.
3. In the Target Air Base window, select the air base to attack.
4. Optionally, in the Part to Target list, select a specific type of target at the air base (buildings, apron, runway, or no specific part).
5. In the Ingress Route window, select the route the unit should follow to approach the air base.
6. In the Egress Route window, select the route the unit should follow to leave the air base.
7. Click OK.

35.2.3 Air-to-Ground Attack

The Air-to-Ground Attack task requires an attack (ingress) route, an exit (egress) route, and a target point.

When the aircraft reaches the target point, it attacks with each type of HE ammunition it has. It expends all ammunition of each type. There is a 30-second delay between attacks with each ammunition type.

➤ **To assign an Air-to-Ground Attack task:**

1. Select the unit that will attack.
2. Choose **Task > Engagement > Air-to-Ground Attack**, or click the Air-to-Ground Attack button (). The Air-to-Ground Attack parameters display.
3. Select a target point.
4. Select an ingress route.
5. Select an egress route.

   This is a multiple list dialog box. For details about how to control which list has focus, see "29.1.7 Specifying Parameters for Tasks" on page 563.

6. Click OK.

35.2.4 Attack by Fire

The Attack By Fire task is for use by units created by aggregating multiple units. When assigned, the units move to the specified location. Armor and mechanized units are placed towards the front of the unit. Artillery is placed in the rear. Other units are placed

between the forward and rear simulation objects. When the unit is in position, artillery begins indirect fire on the target and the other units engage any enemy simulation objects that are in range.

➤ **To assign an Attack By Fire task:**

1. Select the unit.
2. Choose **Task > Engagement > Attack By Fire**. The Attack By Fire parameters display.
3. Specify the location of the target entity.
4. Specify a location for the attacking unit.
5. Click OK.

35.2.5 Attack with Anti-Air Missile

This task is available to any simulation object that has anti-air missiles.

➤ **To assign an Attack with Anti-Air Missile task:**

1. Select the simulation object that will attack.
2. Choose **Task > Engagement > Attack with Anti-Air Missile**, or click the Attack with Anti-Air Missile button (). The Attack with Anti-Air Missile parameters display.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

35.2.6 Attack with Anti-Ship Missile

This task is available to any simulation object that has anti-ship missiles.

➤ **To assign an Attack with Anti-Ship Missile task:**

1. Select the simulation object that will attack.
2. Choose **Task > Engagement > Attack with Anti-Ship Missile**, or click the Attack with Anti-Ship Missile button (). The Attack with Anti-ship Missile parameters display.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

35.2.7 Attack with Land-Attack Missile

This task is available to entities that have land-attack missiles, such as the Arleigh Burke DDG.

➤ **To assign an Attack with Land-Attack Missile task:**

1. Select the simulation object that will attack.
2. Choose **Task > Engagement > Attack with Land-Attack Missile**, or click the Attack with Land-Attack Missile button (). The Attack with Land-Attack Missile parameters display.
3. Select a target.
4. Specify the number of missiles to fire.
5. Click OK.

35.2.8 Attack with Torpedo

This task is available to entities that have torpedoes, such as the Arleigh Burke DDG.

➤ **To assign an Attack with Torpedo task:**

1. Select the entity that will attack.
2. Choose **Task > Engagement > Attack with Torpedo**, or click the Attack with Torpedo button (). The Attack with Torpedo parameters display.
3. Select a target.
4. Specify the number of torpedoes to fire.
5. Click OK.

35.2.9 Automatic Air Defense

When a simulation object has the Automatic Air Defense task, it automatically fires at enemy aircraft that enter the specified area. A simulation object must have an anti-air weapon, such as air-to-air missile to execute this task.

➤ **To assign an Automatic Air Defense task:**

1. Select a simulation object that has anti-air capability.
2. Choose **Task > Engagement > Automatic Air Defense**, or click the Automatic Air Defense button (). The Automatic Air Defense parameters display.
3. Select the area in which the simulation object should automatically fire anti-air weapons.
4. Click OK.

35.2.10 Biological Attack

The Biological Attack task lets a simulation object create an area contaminated with biological agents.

When you assign a Biological Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the simulation object is specified in the biological weapon's weapon system. You can edit these values in the Simulation Object Editor.

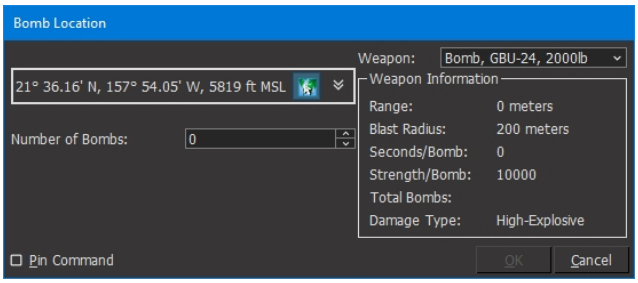
➤ **To assign a Biological Attack task:**

1. Select the simulation object that will make the attack.
2. Choose **Task > Engagement > Biological Attack**. The Biological Attack parameters display.
3. Click a location for the center of the area of attack or specify the coordinates.
4. In the Agent Type list, select the type of biological agent.
5. Click OK.

35.2.11 Bomb Location

You can specify that a simulation object bomb a location. When you choose a weapon, the Bomb Location dialog box displays information about the chosen weapon (Figure 253).

Figure 253. Bomb Location dialog box



➤ **To assign a Bomb Location task:**

1. Select the simulation object that will attack.
2. Choose **Task > Engagement > Bomb Location**. The Bomb Location parameters display.
3. Click the location to bomb or enter the coordinates. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Specify the number of bombs to drop.
5. In the Weapon list, select the type of bomb to drop.
6. Click OK.

35.2.12 Chemical Attack

The Chemical Attack task lets a unit create an area contaminated with chemical agents.

When you assign a Chemical Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the unit is specified in the chemical weapon's weapon system. You can edit this value in the Simulation Object Editor.

➤ **To assign a Chemical Attack task:**

1. Select the unit that will make the attack.
2. Choose **Task > Engagement > Chemical Attack**. The Chemical Attack parameters display.
3. Click a location for the center of the area of attack or specify the coordinates.
4. In the Agent Type list, select the type of chemical agent.
5. Click OK.

35.2.13 Indirect Fire

The Indirect Fire task allows a simulation object to fire on a point. The indirect fire scatters bursts within a sheaf radius, which is a circle around the target point.

If multiple units are within the radius, they are all attacked. Attack strength is applied fully to each attacked simulation object. It is not divided among them. For example, if there are four units in the sheaf radius and the attack strength is 100, an attack strength of 100 is applied to each of the units.

If the blast effect radius of the munitions being fired is less than the sheaf radius, the attack strength is reduced by the ratio of the areas of the blast circle to the sheaf circle. For example, if the sheaf radius is 100 and the blast effect radius is 50, the ratio of the areas is 1:4, so the attack strength would be 25% of the full value.

➤ **To specify indirect fire for a unit:**

1. Select the unit.
2. Choose **Task > Engagement > Indirect Fire**, or click the Indirect Fire button (). The Indirect Fire parameters display.
3. Select the point on which you want to center the indirect fire.
4. In the Weapon list, select the weapon you want to use. Its characteristics are displayed.
5. In the Number of Attacks box, specify how many attacks to make.
6. In the Sheaf Radius box, specify the radius, in meters, around which to scatter indirect fire. This value must be equal to or less than the Max Sheaf Radius value shown in the Weapon Information group box for the selected weapon.
7. Click OK.

35.2.14 Nuclear Attack

The Nuclear Attack task lets a simulation object create an area contaminated with nuclear radiation.

When you assign a Nuclear Attack task, you specify the center of the area in which it will take place. The size of the area and the allowed range from the simulation object is specified in the nuclear weapon's weapon system. You can edit this value in the Simulation Object Editor.

➤ **To execute a nuclear attack:**

1. Select the simulation object that will make the attack.
2. Choose **Task > Engagement > Nuclear Attack**. The Nuclear Attack parameters display.
3. Click a location for the center of the attack area or specify the coordinates.
4. In the Agent Type list, select the type of radiation.
5. Click OK.

35.3. Combat Engineering Tasks

The following topics describe tasks in the combat engineering category that apply to aggregate-level scenarios.

35.3.1 Breach Obstacles

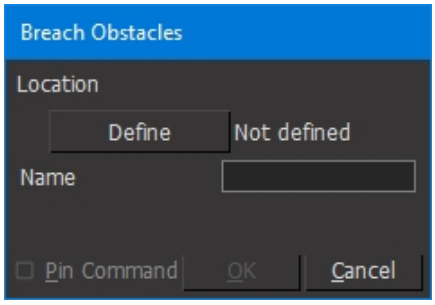
The Breach Obstacles task lets an engineering unit breach an obstacle so that units can pass through it. Depending on the breaching system a unit has, it may be able to breach fortifications, minefields, or ditches.

A breach line is specified with two points.

➤ **To assign a Breach Obstacles task:**

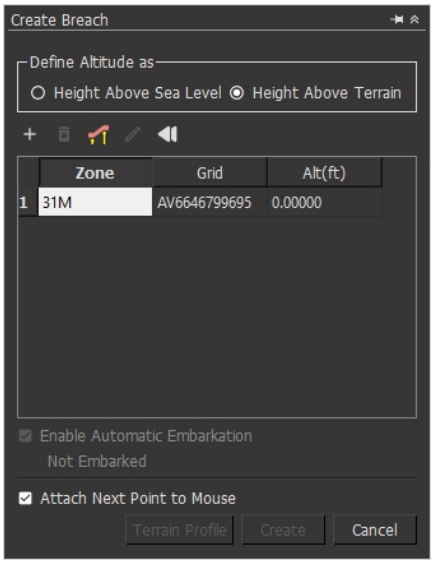
1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Breach Obstacles**, or click the Breach Obstacles button (). The Breach Obstacles parameters display (Figure 254).

Figure 254. Breach Obstacles dialog box



3. Click Define. The cursor changes to draw mode and the Create Breach dialog box opens or becomes available (Figure 255).

Figure 255. Create Obstacles dialog box



4. Draw a two-point line to specify where to create the breach. After the second click the line disappears. This is because it is not being created now. The Breach Obstacles dialog box button now says Edit and the text next to it says Defined.
5. Click OK.

35.3.2 Construct Abatis

Combat engineering units can construct abatises. An abatis is specified as a point.

> You can create abatises directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To assign a Construct Abatis task:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Abatis**, or click the Construct Abatis button (). The Construct Abatis parameters display (similar to Figure 254 on the previous page). The Location is specified as "Not defined".
3. Click Define. The Create Abatis dialog box opens (similar to Figure 255 above).
4. Click where you want to locate the abatis.
5. In the Create Abatis dialog box, click OK.
6. Optionally, specify a name for the abatis.
7. In the Construct Abatis dialog box, click OK.

35.3.3 Construct Anti-Tank Ditch

Combat engineering units can construct anti-tank ditches. An anti-tank ditch is a linear object. This task takes place over time.

> You can create anti-tank ditches directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To assign a Construct Anti-Tank Ditch task:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Anti-Tank Ditch**, or click the Construct Anti-Tank Ditch button (). The Construct Anti-Tank Ditch parameters display (similar to Figure 254). The Location is specified as "Not defined".
3. Click Define. The Create Anti-Tank Ditch dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertexes of the line as you would when creating any linear object, or add points in the dialog box and click Create.
5. Optionally, specify a name for the ditch.
6. Click OK.

35.3.4 Construct Barbed Wire

Combat engineering units can construct barbed wire obstacles. Barbed wire is specified as a line.

> You can create barbed wire directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To construct barbed wire:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Barbed Wire**, or click the Construct Barbed Wire button (). The Construct Barbed Wire parameters display (similar to Figure 255 on the previous page). The Location is specified as "Not defined".
3. Click Define. The Create Barbed Wire dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the barbed wire.
6. In the Construct Barbed Wire dialog box, click OK.

35.3.5 Construct Barricade

Combat engineering units can construct barricades. A barricade is specified as a line.

> You can create barricades directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To construct a barricade:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Barricade**, or click the Construct Barricade button (). The Construct Barricade parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Barricade dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the barricade.
6. In the Construct Barricade dialog box, click OK.

35.3.6 Construct Berm

Combat engineering units can construct berms. A berm is specified as a line.

> You can create berms directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To construct a berm:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Berm**, or click the Construct Berm button (). The Construct Berm parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Berm dialog box opens.
4. Click to specify the vertexes of the line as you would when creating any linear object.
5. Optionally, specify a name for the berm.
6. In the Construct Berm dialog box, click OK.

35.3.7 Construct Booby Trap

Combat engineering entities can construct booby traps. A booby trap is a location where explosives detonate if triggered by the presence of a simulation object. Booby traps cause some level of attrition on the affected simulation object. To create a booby trap, a unit must have explosive resources.

You can create booby traps directly from the Create Object Palette. This task is used when you want a unit to construct this object.

➤ **To construct a booby trap:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Booby Trap**, or click the Construct Booby Trap button (). The Construct Booby Trap parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Booby Trap dialog box opens.
4. Click where you want to locate the booby trap or type the coordinates in the dialog box and click OK.
5. Optionally, specify a name for the booby trap.
6. Click OK.

35.3.8 Construct Bridge

Combat engineering units can construct bridges. A bridge is specified as a two-point line.

You can create bridges directly from the Create Object Palette. This task is used when you want a unit to construct this object.

➤ **To construct a bridge:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Bridge**, or click the Construct Bridge button (). The Construct Bridge parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Bridge dialog box opens and the cursor changes to draw mode.
4. Click on the terrain to specify the start and end vertices of the bridge. The Create Bridge dialog box closes.
5. Optionally, specify a name for the bridge.
6. Click OK.

35.3.9 Construct Dry Gap

Combat engineering entities can construct dry gaps. A dry gap is a type of ditch. (Use Improve Ditch and Destroy Ditch to continue building a dry gap or destroy it.)

You can create dry gaps directly from the Create Object Palette. Use this task when you want a unit to construct this object.

➤ **To construct a dry gap:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Dry Gap**, or click the Construct Dry Gap button (). The Construct Dry Gap parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Dry Gap dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertices of the dry gap.
5. Double-click to finish the dry gap, or click Create in the Create Dry Gap dialog box.
6. Optionally, specify a name for the dry gap.
7. Click OK.

35.3.10 Construct Fortified Area

Combat engineering entities can construct fortified areas. A fortified area is defined as a box. You specify the southwest corner of the area. Construction of a fortified area takes place over time.

ⓘ You can create fortifications directly from the Create Object Palette. This task is used when you want a unit to construct this object.

➤ **To construct a fortified area:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Fortified Area**, or click the Construct Fortified Area button (). The Construct Fortified Area parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Fortified Area dialog box opens and the cursor changes to show an area.
4. Click where you want to locate the southwest corner of the area or type the coordinates in the dialog box and click Create.
5. Optionally, specify a name for the area.
6. Click OK.

35.3.11 Construct Fortified Line

Combat engineering entities can construct fortified lines.

ⓘ You can create fortifications directly from the Create Object Palette. This task is used when you want a unit to construct this object.

➤ **To construct a fortified line:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Fortified Line**, or click the Construct Fortified Line button (). The Construct Fortified Line parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Fortified Line dialog box opens and the cursor changes to draw mode.
4. Click to specify the vertexes of the line as you would when creating any linear object, or add points in the dialog box and click Create.
5. Optionally, specify a name for the line.
6. Click OK.

35.3.12 Construct Minefield

Combat engineering entities can construct volcano minefields. Minefields are fixed size rectangles.

> You can create minefields directly from the Create Object Palette. This task is used when you want a unit to construct this obstacle.

➤ **To construct a minefield:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Minefield**, or click the Construct Minefield button (). The Construct Minefield parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Minefield (Volcano) dialog box opens and the cursor changes to show a minefield.
4. Click where you want to locate the southwest corner of the minefield or type the coordinates in the Create Minefield (Volcano) dialog box and click Create.
5. Optionally, specify a name for the minefield.
6. In the Construct Minefield dialog box, click OK.

35.3.13 Construct Strong Point

Combat engineering entities can construct strong points. A strong point is defined as a box. You specify the southwest corner.

> You can create strong points directly from the Create Object Palette. This task is used when you want a unit to construct this object.

➤ **To construct a strong point:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Construct Strong Point**, or click the Construct Strong Point button (). The Construct Strong Point parameters display (similar to Figure 255 on page 720). The Location is specified as "Not defined".
3. Click Define. The Create Strong Point dialog box opens and the cursor changes to show a box.
4. Click where you want to locate the southwest corner of the box or type the coordinates in the dialog box and click Create.
5. Optionally, specify a name for the strong point.
6. Click OK.

35.3.14 Destroy Bridge

When you task an engineering unit to destroy a bridge, it consumes its Explosives resource. As the unit attacks the bridge, the bridge's completeness percentage decreases. When the completion percentage reaches zero, the bridge is removed from the scenario.

➤ **To destroy a bridge:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Destroy Bridge**, or click the Destroy Bridge button (). The Destroy Bridge parameters display.
3. Select the bridge you want to destroy.
4. Click OK.

35.3.15 Destroy Ditch

When you task an engineering unit to destroy a ditch or dry gap, it consumes its Explosives resource. As the unit attacks the ditch, the ditch's completeness percentage decreases. When the completion percentage reaches zero, the ditch is removed from the scenario.

➤ **To destroy a ditch:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Destroy Ditch**, or click the Destroy Ditch button (). The Destroy Ditch parameters display.
3. Select the ditch you want to destroy.
4. Click OK.

35.3.16 Destroy Explosive

The Destroy Explosive task lets a unit destroy a booby trap or unexploded ordnance.

➤ **To destroy explosives:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Destroy Explosive**, or click the Destroy Explosive button (). The Destroy Explosive parameters display.
3. Select the booby trap or unexploded ordnance you want to destroy.
4. Click OK.

35.3.17 Destroy Fortification

When you task an engineering unit to destroy a fortification, it consumes its Explosives resource. As the unit attacks the fortification, the fortification's completeness percentage decreases. When the completion percentage reaches zero, the fortification is removed from the scenario.

➤ **To destroy a fortification:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Destroy Fortification**, or click the Destroy Fortification button (). The Destroy Fortification parameters display.
3. Select the fortification you want to destroy.
4. Click OK.

35.3.18 Destroy Obstacle

The Destroy Obstacle task lets a unit destroy obstacles (such as berms, barricades, and barbed wire) that do not have a specific destroy task, such as Destroy Fortification or Destroy Explosive.

➤ **To destroy an obstacle:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Destroy Obstacle**, or click the Destroy Obstacle button (). The Destroy Obstacle parameters display.
3. Select the obstacle you want to destroy.
4. Click OK.

35.3.19 Improve Booby Trap

The Improve Booby Trap task lets a combat engineering unit improve an incomplete booby trap. When you improve a booby trap, it is improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a booby trap.

➤ **To improve a booby trap:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Booby Trap**. The Improve Booby Trap parameters display.
3. Select the booby trap that you want to improve.
4. Click OK.

35.3.20 Improve Breach

The Improve Breach task lets a combat engineering unit improve an incomplete breach. When you improve a breach, it is improved to as complete a strength level as the unit is able to do. You can assign this task to a unit to assist another unit that is breaching an obstacle.

➤ **To improve a breach:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Breach**. The Improve Breach parameters display.
3. Select the breach you want to improve.
4. Click OK.

35.3.21 Improve Bridge

The Improve Bridge task lets a combat engineering unit improve an incomplete bridge. When you improve a bridge, it is improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a bridge.

➤ **To improve a bridge:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Bridge**. The Improve Bridge parameters display.
3. Select the bridge you want to improve.
4. Click OK.

35.3.22 Improve Ditch

The Improve Ditch task lets a combat engineering unit improve an incomplete anti-tank ditch or dry gap. When you improve a ditch, it is improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a ditch.

➤ **To improve a ditch:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Ditch**. The Improve Ditch parameters display.
3. Select the ditch that you want to improve.
4. Click OK.

35.3.23 Improve Fortification

The Improve Fortification task lets a unit improve an incomplete fortification area, line, or strong point. When you improve a fortification, it is improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is creating a fortification.

➤ **To improve a fortification:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Fortification**. The Improve Fortification parameters display.
3. Select the fortification that you want to improve.
4. Click OK.

35.3.24 Improve Obstacle

The Improve Obstacle task lets a unit improve an incomplete obstacle (such as berms, barricades, and barbed wire) that does not have a dedicated improve task, such as Improve Breach or Improve Bridge. When you improve an obstacle it is improved to as complete a strength level as the unit can bring it to. This task can be assigned to a unit to assist another unit that is constructing an obstacle.

➤ **To improve an obstacle:**

1. Select a combat engineering unit.
2. Choose **Task > Combat Engineering > Improve Obstacle**. The Improve Obstacle parameters display.
3. Select the obstacle that you want to improve.
4. Click OK.

35.4. Air Base Tasks

The following topics describe tasks for aggregate-level air bases.

35.4.1 Launch Flight

An air base uses the Launch Flight task to configure and launch a flight of aircraft. A flight can be launched to execute a single unit mission or it can be one of several flights being launched as part of a multi-unit mission. If you want to set up a multi-unit mission, you must create a unit independently of the air base to act as the mission superior. For information about missions, see "27.6. Air Missions" on page 542.

When you assign this task, aircraft of the indicated type are taken from the available aircraft at the base. The air base determines the munitions and fuel required by the flight given the loadout and provides them from its stores. Air crews are supplied from the available air crews. If the air base does not have enough of any of these resources, the task fails and the flight is not scheduled to launch.

When the flight is launched, the aircraft and crews are removed from the air base.

A Launch Flight task takes values for these parameters:

- Call sign.
- Aircraft type.
- Number of aircraft in the flight.
- Loadout.
- Mission superior.
- Mission type. The mission can be one of:
  - BAI. Battlefield air interdiction.
  - CAP. Combat air patrol.
  - CAS. Close air support.
  - Escort.
  - Intercept.
  - OCA. Offensive counter air (bombing air bases).
  - Recon. Reconnaissance.
  - SEAD. Suppression of enemy air defense.
  - Sweep. Sweep enemy aircraft from the air space where you are going to attack.
  - Tanker.

  The mission type is informational. It does not influence the behavior of the flight.
- Launch time. This can be immediate or at a specified date and time.
- Wait for launch. Global plans only. If selected, the global plan does not move to the next task until the flight is launched.

➤ **To launch a flight:**

1. Select an air base.
2. Choose **Task > Launch Flight**, or click the Launch Flight button (). The Launch Flight parameters display.
3. Specify a call sign for the flight. It can be any text.
4. Select the aircraft type. It must be a type of aircraft that is already assigned to the air base. The list of aircraft types may include aircraft that are not assigned.
5. Specify the number of aircraft in the flight.
6. The Unit Type is not editable. It is taken from the aircraft type.
7. Select a loadout from the Loadout list.
8. Optionally, specify a mission superior if this flight is subordinate to another one.
9. In the Mission Type list, select the type of mission this is.
10. Select Launch as Soon as Aircraft are Available or specify a date and time for the launch.
11. Optionally, for global plans only, specify that the global plan not proceed to the next task until the flight is launched (Wait for Launch check box).

35.4.2 Launch QRA Aircraft

The Launch QRA Aircraft task creates an air unit and launches the specified number of aircraft that are on Quick Reaction Alert (QRA) status.

➤ **To launch a QRA unit:**

1. Select an air base.
2. Choose **Task > Launch QRA Aircraft**, or click the Launch QRA Aircraft button. The Launch QRA Aircraft parameters display.
3. Specify a call sign for the unit that is to be created.
4. Specify the number of aircraft to include in the unit.
5. Click OK.

35.4.3 Prepare QRA

The Prepare QRA task assigns aircraft to quick reaction alert (QRA) status. You specify the type of aircraft, loadout, when they should be prepared, and their response time. Once the aircraft are prepared they are available to be launched.

➤ **To prepare aircraft for QRA status:**

1. Select an air base.
2. Choose **Task > Prepare QRA**, or click the Prepare QRA button (). The Prepare Aircraft for QRA Status parameters display.
3. In the Aircraft Type list, select an aircraft type.
4. In the Number list, select the number of aircraft.

5. In the Loadout list, select a loadout.
6. If you want aircraft to be prepared as soon as they are available, select the Prepare As Soon As Aircraft Are Available check box. If you want them prepared at a specific time, clear the check box and in the Start Time box, specify the time when you want the preparation process to start.
7. Select a Response Time option.
8. Click OK.

35.5. Movement Tasks for Aggregate-Level Scenarios

The following topics describe tasks in the movement category that apply to aggregate-level scenarios.

35.5.1 Change MOPP Level

Units automatically raise their mission oriented protective posture (MOPP) level when they encounter a contamination area. However, you can manually change a unit's MOPP level.

Supported MOPP levels are:

- MOPP Level 0 — Protective mask carried. Suit, gloves, and boots accessible/available.
- MOPP Level 1 — Suit worn. Mask, gloves and boots carried.
- MOPP Level 2 — Suit and boots worn. Gloves and mask carried.
- MOPP Level 3 — Suit, boots and mask worn. Gloves carried.
- MOPP Level 4 — All protection worn.

Changes to MOPP level do not take place immediately. (To change the MOPP level immediately, see "41.27. MOPP Level" on page 922 for details on the set data request.) Increasing the MOPP level degrades a simulation object's speed and combat effectiveness. Increasing the MOPP level does not remove existing contamination, but prevents most NBC contamination from harming the simulation object.

➤ **To assign a Change MOPP Level task:**

1. Select the unit.
2. Choose **Task > Movement > Change MOPP Level**, or click the Change MOPP Level button (). The Change MOPP Level parameters display.
3. Specify a new MOPP level.
4. Click OK.

35.5.2 Change Posture

You can set a simulation object's posture to:

- Travel
- Reconnaissance
- Hasty attack
- Deliberate attack
- Hasty defense
- Deliberate defense

Changes to posture take place over time. To change posture immediately, use Set Posture.

➤ **To assign a Change Posture task:**

1. Select the simulation object.
2. Choose **Task > Movement > Change Posture**, or click the Change Posture button. The Change Posture parameters display.
3. Specify a new posture.
4. Click OK.

35.5.3 Escort Air Unit

The Escort Air Unit task causes an aircraft unit to move to a location and escort the specified unit.

➤ **To assign an escort air unit task:**

1. Select an air unit.
2. Choose **Task > Movement > Escort Air Unit**, or click the Escort Air Unit button (). The Escort Air Unit parameters display.
3. Select a point as the location to meet.
4. Select the unit to escort.
5. Click OK.

35.5.4 Halt Movement

The Halt Movement task causes a simulation object to stop moving.

➤ **To halt a simulation object's movement:**

1. Select the simulation object.
2. Choose **Task > Movement > Halt Movement**, or click the Halt Movement button (). Movement stops immediately.

35.5.5 Land at Air Base

The Land at Air Base task causes an air unit to request an air base to let it land. The air unit tells the air base how many aircraft it has and how long a runway it requires.

If the request is accepted, the unit flies to the air base and requests final permission to land. The air base absorbs the resources of the unit and deletes it from the scenario. For complete details about how landing works, see "27.8.2 Landing Air Units" on page 547.

➤ **To land an aircraft unit at an air base:**

1. Select an aircraft unit.
2. Choose **Task > Movement > Land at Air Base**, or click the Land at Air Base button ( ). The Land at Air Base parameters display.
3. Select the air base you want the unit to land at.
4. Click OK.

35.5.6 Move

The Move task simply directs a ground unit to move to a location or waypoint. Optionally, you can specify that the unit follow roads.

➤ **To assign the Move task:**

1. Select the simulation object.
2. Choose **Task > Movement > Move**. The Move parameters display.
3. Select the destination:
   - Select the Use Location option and then click the location to move to or enter the coordinates.
   - Select the Use Waypoint option and then specify waypoint or point tactical graphic.
4. Optionally, select the Use Roads check box.
5. Click OK.

35.5.7 Move Along Route

For details, please see "30.24. Move Along Route" on page 599.

35.5.8 Move Along Route Retrograde

Move Along Route Retrograde is the same as Move Along Route, except that the simulation object does not change its heading to the direction of movement. It maintains the heading at the start of movement or a heading necessary for defensive operations. See "30.24. Move Along Route" on page 599 for the procedure for this task.

35.5.9 Move To

For details, please see "30.28. Move To" on page 604.

35.5.10 Move to Retrograde

Move to Retrograde is the same as Move To, except that the simulation object does not change its heading to the direction of movement. It maintains the heading at the start of movement or a heading necessary for defensive operations. Please see "30.28. Move To" on page 604 for the procedure for this task.

35.5.11 Move to Location (Plan Along Roads)

The Move to Location (Plan Along Roads) task directs a ground unit to move to a location using roads, if there is a road network.

When a ground unit receives this task, it determines whether there is a road network nearby. If there is, it moves directly to the nearest point on the road network and then moves along the roads until it reaches the point on the road that is closest to the destination. At that point, the unit moves off the road and proceeds directly to the destination.

➤ **To assign a Move to Location (Plan Along Roads) task:**

1. Select the simulation object.
2. Choose **Task > Movement > Move to Location (Plan Along Roads)**. The Move to Location (Plan Along Roads) parameters display.
3. Click on the terrain where you want the entity to move, or enter the coordinates of the location.
4. Optionally, you can select the Display Route check box to visualize the unit's planned route.
5. Click OK.

35.5.12 Move to Waypoint (Plan Along Roads)

The Move to Waypoint (Plan Along Roads) task directs a ground unit to move to a waypoint and use the road network if it is available.

When a ground unit receives this task, it determines whether there is a road network nearby. If there is, it moves directly to the nearest point on the road network and then moves along the roads until it reaches the point on the road that is closest to the destination. At that point, the unit moves off the road and proceeds directly to the destination.

➤ **To assign a Move to Waypoint (Plan Along Roads) task:**

1. Select the simulation object.
2. Choose **Task > Movement > Move to Waypoint (Plan Along Roads)**. The Move to Waypoint (Plan Along Roads) parameters display.
3. Select the waypoint.
4. Optionally, you can select the Display Route check box to visualize the unit's planned route.
5. Click OK.

35.5.13 Patrol Route

For details, see "30.38. Patrol Route" on page 609.

35.5.14 Patrol Between

For details, please see "30.37. Patrol Between" on page 609.

35.5.15 Turn To Heading

The Turn to heading task for aggregate-level scenarios causes entities to turn to a new heading. It happens instantly, similar to the Heading set data request.

➤ **To assign a Turn to Heading task:**

1. Select the entity.
2. Choose **Task > Movement > Turn to Heading**. The Turn to Heading parameters display.
3. Type a heading.
4. In the Relative To list, select one of these options:
   - True North. Turn to the specified absolute heading (true north).
   - Host Object (if embarked). If embarked, turn to the specified heading relative to the host simulation object.
   - Current Heading. Turn the specified number of degrees relative to the current heading.
   - Magnetic North. Turn to the specified heading relative to magnetic north.
5. Click OK.

35.5.16 Wait for Escort

You can instruct an aircraft unit to move to a waypoint and wait for an escort.

➤ **To assign a Wait for Escort task:**

1. Select an aircraft unit.
2. Choose **Task > Movement > Wait for Escort**, or click the Wait for Escort button (). The Wait for Escort parameters display.
3. Select the waypoint that the unit should move to.
4. Click OK.

35.5.17 Wait Tasks

You can specify unconditional wait, wait for a duration, or wait until a simulation time. For details, please see "34.6. Wait Tasks" on page 697.

35.6. Embarkation Tasks

The following topics describe tasks in the embarkation category that apply to aggregate-level scenarios.

35.6.1 Disembark

For details, please see "34.1.1 Disembark" on page 681.

35.6.2 Disembark All

For details, please see "34.1.2 Disembark All" on page 681.

35.6.3 Embark

For details, please see "34.1.3 Embark" on page 681.

35.7. Radio Tasks

This section lists the tasks that allow you to use the VR-Forces radio network to send messages, tasks, and set data requests.

35.7.1 Send Radio Set

For details, please see "34.5.1 Send Radio Set" on page 695.

35.7.2 Send Radio Task

For details, please see "34.5.2 Send Radio Task" on page 696.

35.7.3 Send Text Message

For details, see "34.5.3 Send Text Message" on page 697.

35.8. Other Tasks

The following topics describe other tasks that apply to aggregate-level scenarios.

35.8.1 Activate Jammer

The Activate Jammer task turns on jamming for a simulation object equipped with a jamming emitter. This task has the same effect as Set Emitter - On.

➤ To activate a simulation object's jammer, choose **Task > Jamming > Activate Jammer**, or click the Activate Jammer button ().

35.8.2 Deactivate Jammer

The Deactivate Jammer task turns off jamming for a simulation object equipped with a jamming emitter. This task has the same effect as Set Emitter - Off.

➤ To deactivate a simulation object's jammer, choose **Task > Jamming > Deactivate Jammer**, or click the Deactivate Jammer button ().

35.8.3 Drop Fuel Tanks

The Drop Fuel Tanks task causes an air unit to drop its fuel tanks.

➤ **To drop an air unit's fuel tanks:**

1. Select an air unit.
2. Choose **Task > Drop Fuel Tanks**, or click the Drop Fuel Tanks button ().

35.8.4 FASCAM

Entities with a FASCAM weapon system, such as artillery, can fire FASCAM (family of scatterable mines).

➤ **To fire FASCAM:**

1. Select a simulation object that can execute this task.
2. Choose **Task > FASCAM**, or click the FASCAM button (). The FASCAM parameters display (similar to Figure 254). The Location is specified as "Not defined".
3. Click Define. The Create Minefield (ADAM-RAAM) dialog box opens and the cursor changes to show a minefield.
4. Click where you want to locate the southwest corner of the minefield or type the coordinates in the Create Minefield (ADAM-RAAM) dialog box and click Create.
5. Optionally, specify a heading.
6. In the FASCAM dialog box, click OK.

35.8.5 Refuel from Tanker

The Refuel from Tanker task causes an aircraft unit to request refueling from a fuel tanker. A message is sent to the tanker indicating the amount of fuel required and the type of connection. The tasked aircraft then performs a series of steps:

1. The aircraft unit flies to a standby position near the tanker and follows the tanker.
2. The aircraft unit follows the tanker at range 0 and refuels.
3. The aircraft unit moves away from the tanker and follows at a small distance.

Each of these steps is performed in response to a message from the tanker indicating that the tanker is ready. During this time the receiving aircraft unit also gives the tanker information about the number of aircraft in the unit and the maximum transfer rate for the aircraft.

If the receiving aircraft unit is an echelon containing subordinate units, then each subordinate coordinates with the tanker independently.

➤ **To refuel an aircraft unit:**

1. Select the unit.
2. Choose **Task > Combat Service Support > Refuel from Tanker** or click the Refuel from Tanker button (). The Refuel from Tanker parameters display.
3. Select the tanker you want to refuel from.
4. Click OK.

35.8.6 Refueling Mission

The Refueling Mission task causes a tanker aircraft to follow a route or move to a waypoint and fly a holding pattern. It awaits requests from other aircraft to refuel; see "35.8.5 Refuel from Tanker" on the previous page.

➤ **To issue a refueling mission:**

1. Select an aircraft that has refueling capability.
2. Choose **Task > Combat Service Support > Refueling Mission**, or click the Refueling Mission button (). The Refueling Mission parameters display.
3. Specify the altitude at which to conduct the refueling mission.
4. Specify how much fuel to reserve for the tanker's own use. If the fuel level drops to this amount, the tanker will stop refueling other aircraft.
5. Specify a route to follow or specify the parameters for a race track holding pattern.
6. Click OK.

35.8.7 Send NBC Report

Sending an NBC report is the equivalent of sending a spot report when an NBC area is found. (An NBC area is a nuclear, biological, or chemical contamination area.) NBC reports are sent automatically if a simulation object has an NBC sensor. However, this task allows you to send a report manually. For more on NBC areas, see "27.1.6 Contamination Areas" on page 532.

➤ **To send an NBC report:**

1. Select a simulation object.
2. Choose **Task > Send NBC Report**. The Send NBC Report parameters display.
3. Select the contamination area for which to send a report.
4. Specify the Engagement Result. This is any text that you want to accompany the report.
5. Click OK.

35.8.8 Send Obstacle Report

Sending an obstacle report is the equivalent of sending a spot report when an obstacle area is found. Obstacle reports are sent automatically if a simulation object has an engineering object sensor. However, this task allows you to send a report manually.

➤ **To send an obstacle report:**

1. Select a simulation object.
2. Choose **Task > Send Obstacle Report**. The Send Obstacle Report parameters display.
3. Select the obstacle for which to send a report.
4. Specify the Engagement Result. This is any text that you want to accompany the report.
5. Click OK.

35.8.9 User Task

For details, please see "34.13. User Task" on page 707.

35.9. Background Process List

Background processes are not described in detail because they run automatically; no user action is required to execute them.

Table 33 lists the background tasks provided. These tasks are only available for units in aggregate-level SMSs.

Table 33. Background processes

Background Process	Description
Attack Enemies	Determines when to begin and stop attacking enemies.
Attrit Air Base	Handles air-to-ground attacks against an air base.
Calculate Maximum Speed Modifier	Calculates a modifier that is applied to the maximum possible speed of a unit to modify its movement rate.
Make EW Attack	If a jammer system is turned on, attack all non-friendly simulation objects within range of the jammer. The simulation object attacks continuously.
Manage Air Base	Launches scheduled flights and updates resources for an air base.
Perform Attrition	Determines when a unit is being attacked and modifies its health based on the current attrition rate.
Provide Supplies	Continually checks to see if conditions allow providing supplies (not in combat, not ordered to stop providing supplies). Checks to see what friendly units are in range, and sends each new unit an influence to let it know that this unit is available. When a unit goes out of range, or this unit is ordered to stop providing supplies, it sends a stop influence to that unit.
Receive Construction	Allows an engineering object to receive construction from other simulation objects.

Table 33. Background processes (continued)

Background Process	Description
Resize Footprints	Updates the size of a simulation object's physical and sensor footprints.
Update Combat Effectiveness	Updates the combat effectiveness state of a simulation object.
Update EW Degradation	Processes jamming interactions and computes the level of degradation in communications and radar from these attacks.
Use Supplies	Uses supplies based on the simulation object's current activities.



36. Creating Scripted Tasks and Sets

VR-Forces lets you create new tasks and set data requests by writing scripts. You do not need to know C++ or have a developers license to write these scripts. This chapter explains how to create and edit script metadata and how to organize and enable scripts. "Chapter 37. Writing Scripts" on page 783 explains how to write script code. "Chapter 39. Using Qt Designer for Script Dialog Boxes" on page 853 explains how to use Qt Designer to lay out dialog boxes.

36.1. Introduction to Scripts 745
- 36.1.1 Scripted Tasks 746
- 36.1.2 Scripted Sets 746
- 36.1.3 Reactive Task Scripts 746
- 36.1.4 Background Process Scripts 746
- 36.1.5 Script Metadata 747
- 36.1.6 The Lua Scripting Language 747

36.2. Creating a New Script 747
- 36.2.1 Specifying the Script ID 753
- 36.2.2 Specifying a Menu Icon 753
- 36.2.3 Specifying an Extended Name 754
- 36.2.4 Specifying a Short Description 754
- 36.2.5 Specifying a Description 754
- 36.2.6 Specifying the Parameters for a Script 755
- 36.2.7 Specifying a Menu Location 764
- 36.2.8 Specifying Action Categories 765
- 36.2.9 Configuring a Script's Availability to a Simulation Object Type 766
- 36.2.10 Specifying Equivalent Scripts for a Toolbar Button 769
- 36.2.11 Specifying the Programming Language for a Script 770
- 36.2.12 Configuring Scripted Tasks to Run Concurrently 771
- 36.2.13 Creating Scripted Sets 771
- 36.2.14 Creating Reactive Tasks 772
- 36.2.15 Creating Background Processes 774

36.3. Saving Scripts 775
36.4. Editing a Script 776
36.5. Filtering the List of Scripts 776
36.6. Organizing Scripts into Folders 777
36.6.1 Adding a Folder 777
36.6.2 Renaming a Folder 777
36.6.3 Deleting a Folder 777
36.6.4 Adding Scripts to a Folder 777
36.6.5 Removing a Script from a Folder 777
36.7. Exporting and Importing Scripts 777
36.7.1 Importing a Script Package 778
36.8. Copying a Script 779
36.9. Deleting a Script 779
36.10. Creating a System Script 779
36.10.1 Including System Scripts on the Task and Set Menus 780
36.11. Specifying a Script Editor 781
36.12. Editing Lua Files 782

36.1. Introduction to Scripts

The script capability in VR-Forces allows you to create new tasks and set data requests for simulation objects without using the VR-Forces Toolkit and C++ API. Scripts use the Lua programming language. The actions that simulation objects can carry out are still limited to the basic C++ tasks provided by VR-Forces. However, since Lua is a full-featured scripting language, it is far more powerful than the language used to create plans and you can be much more creative in how you use the basic tasks. In addition to functions for tasks and sets, Lua scripts can make many types of calls to the simulation engine to get simulation object status or other simulation data. Scripts can also set some simulation data.

Scripts are classified as system scripts or as scenario scripts. System scripts are saved as part of a simulation model set (SMS) and are available to all scenarios that use that SMS. Scenario scripts are available only in the scenario for which the script was created. However, you can save a scenario script as a system script if you want to and, just as with other data used by VR-Forces, there are ways of copying scripts from one VR-Forces installation to another.

VR-Forces supports the following types of scripts:

- Task. Scripted tasks can be listed on the Task menu and used as independent tasks or in plans just like C++ tasks.
- Reactive task. A reactive task monitors the simulation and activates when the conditions set in the script become true. Reactive tasks are enabled or disabled for simulation objects through the GUI.
- Background process. A background process runs continuously from the time a simulation object is created. There is no user interface for assigning background processes to simulation objects. Their use is based entirely on the scripting process.
- Set data request. Set data request scripts change simulation object state without interrupting the current task.

From a Lua script you can:

- Create and delete simulation objects and tactical graphics.
- Set weather and environment state.
- Get simulation object state, such as:
  - Aggregation
  - Location
  - Embarkation
  - Type
  - Force
  - Damage state
  - Velocity
- Monitor and set simulation object resources.

- Get simulation object sensor contacts.
- Task other simulation objects.
- Subtask self.
- Send Set Data Request messages.
- Test terrain:
  - Get terrain height.
  - Check line of sight.
  - Find feature data.

VR-Forces includes ZeroBrane Studio, a lightweight IDE, as the default script editor. You can configure VR-Forces to use a different editor if you want to. For details, see "36.11. Specifying a Script Editor" on page 781.

36.1.1 Scripted Tasks

Scripted tasks can be added to the Task menu and are treated just like the tasks that are built into VR-Forces.

When you create a scripted task, you can specify input parameters. VR-Forces can automatically build a dialog box for users to specify the parameter values or you can use Qt Designer to lay out the dialog box. For more information, see "Chapter 39. Using Qt Designer for Script Dialog Boxes" on page 853.

36.1.2 Scripted Sets

Scripted sets can be added to the Set menu and are treated just like the set data requests that are built into VR-Forces.

When you create a scripted set, you can specify input parameters. VR-Forces can automatically build a dialog box for users to specify the parameter values or you can use Qt Designer to lay out the dialog box. For more information, see "Chapter 39. Using Qt Designer for Script Dialog Boxes" on page 853.

A scripted set cannot issue tasks to the entity that executes it or to any other entity.

36.1.3 Reactive Task Scripts

Reactive tasks monitor a simulation and activate when the conditions set in the script become true. Reactive tasks are not added to the Task menu. In most respects the process for creating and writing reactive tasks is the same as creating and writing all scripts. Assume that all details about scripts apply to reactive tasks unless noted otherwise. For conceptual details about reactive tasks, see "29.5. Reactive Tasks" on page 572.

36.1.4 Background Process Scripts

The background process is a script that runs in the background for the simulation object types for which it was written. Since a background process has no user interface, it does not have any input parameters. When you create a background process, you specify the simulation object types for which it is available and you write the script.

36.1.5 Script Metadata

A script has two major components, the script code, and the script metadata. The metadata provides all of the information that VR-Forces needs to store the script, add it to a menu, and generate a dialog box.

36.1.6 The Lua Scripting Language

Lua is a lightweight scripting language used by VR-Forces to implement the script feature. VR-Forces includes a set of Lua functions that map the built-in tasks and set data requests as well as the ability to obtain simulation object state information. You use these functions as the starting point for building new tasks. For details about the Lua language, please visit the Lua website (https://www.Lua.org).

36.2. Creating a New Script

To create a new script, you specify the metadata for the script and write the script code. A script's metadata specifies the script's name, where it will be placed on a menu, and its parameters, if any. Specifying metadata may be an iterative process, particularly the parameters, which may change as you write the code for the script. However, the minimum first step in creating a script is to specify a name for the script and add it to the list of scripts. Once you do this, you can edit the metadata and the code as needed until you have completed the script to your satisfaction.

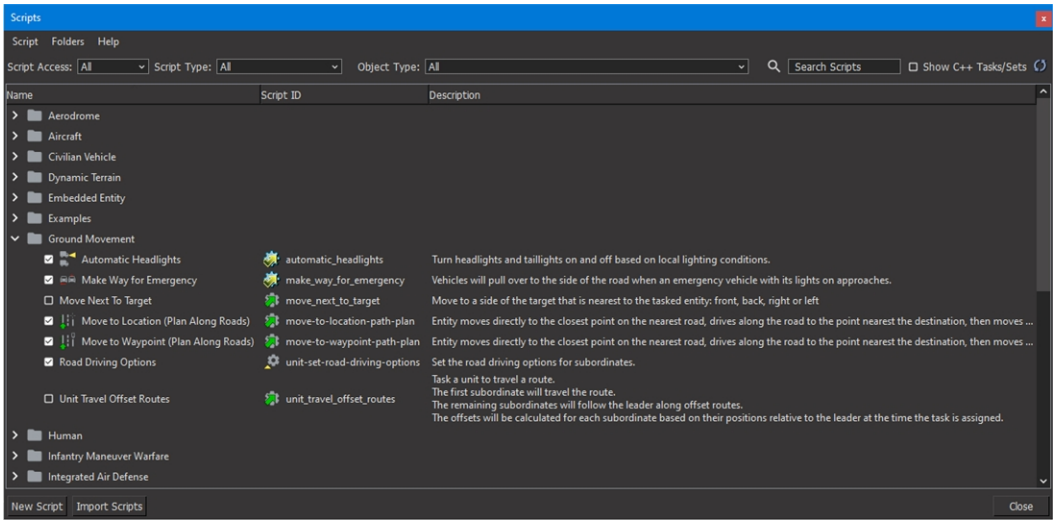
All new scripts are created as scenario scripts. If you want to create a new system script, you save the scenario script as a system script. For details, see "36.10. Creating a System Script" on page 779.

> To create a script, you must have a scenario open. This is true even if you plan to create a system script rather than a scenario script.

➤ **To create a new script:**

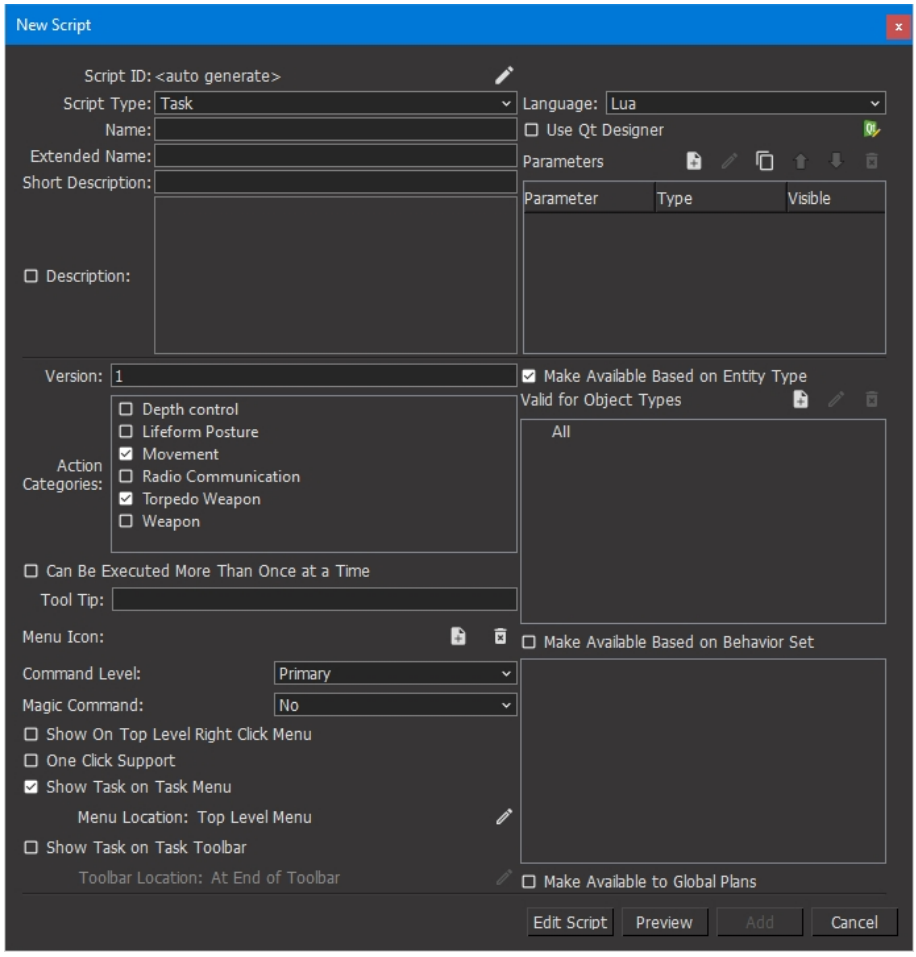
1. Open a scenario. (If you want to create a system script, you must create it in the context of a scenario. Then you will promote it to a system task.)
2. Choose **Simulation > Scripts**. The Scripts dialog box opens (Figure 256).

Figure 256. Scripts dialog box



3. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, see "36.6. Organizing Scripts into Folders" on page 777.)
4. Choose **Script > New Script** or click the New Script button. The New Script dialog box opens (Figure 257).

Figure 257. New Scripted Task dialog box



5. In the Script Type list, select the type of script you are creating (Task, Reactive Task, Set, or Background Process). The dialog box displays the correct set of metadata for the type of script you chose.
6. Complete the fields in the dialog box. They are described in Table 34 and in following sections.
7. Optionally, click Preview to see the dialog box that the script will create.
8. Optionally, click Edit Script to begin writing the code for this script.
9. Click Add. The script is added to the list in the Scripts window.
10. To save the script task as part of the scenario, save the scenario.

> ⚠ If it is a scenario script, it is not saved until you save the scenario.

11. Write the code for the script. For details, see "Chapter 37. Writing Scripts" on page 783.

Table 34. Script metadata

Input Field	Description
Script ID	A unique name for the script. VR-Forces generates a default name, but you will probably want to change it to something meaningful. It is your responsibility to make sure that it is unique. See "36.2.1 Specifying the Script ID" on page 753 for details.
Script Type	One of: • Task. • Reactive Task. • Background Process. • Set. • Task (Behavior). The fields on the New Script dialog box vary based on which type of task is being created. (Behavior tasks involve advanced development; see "Chapter 38. Writing Advanced Scripts with the Behavior Engine" on page 815.)
Language	Specify Lua if you are writing a Lua script. Specify C++ if you have written a new task controller and want to use the script interface to create the GUI portion of the task. (Does not apply to reactive tasks and background processes.) For more information, see "36.2.11 Specifying the Programming Language for a Script" on page 770.
Name	The text that will be added to the Task or Set menu for this script.
Extended Name	An alternative name to use on context-insensitive task or set lists if more than one task or set has the same name. For details about extended names, see "36.2.3 Specifying an Extended Name" on page 754.
Short Description	The text to use for this script in a Plan window.
Description	A description of what the script does. This text is displayed on the dialog box. Providing a description is optional. Also, optionally you can choose to include the description on the task or set dialog box.
Version	Script version. Available as a parameter to a Lua script. For use by script writers who want to have different versions of a script.
Action Categories	For tasks, specifies the categories of tasks that this scripted task will interrupt. (In other words, action categories manage task concurrency. For details about concurrent tasks, see "29.1.4 Concurrent Task Execution" on page 561.) (Does not apply to background processes or sets.)

Table 34. Script metadata (continued)

Input Field	Description
Can Be Executed More Than Once at a Time	Allows an entity to manage more than one instance of this task at a time. For details, see "36.2.12 Configuring Scripted Tasks to Run Concurrently" on page 771.
Tool Tip	Text for a tooltip when the script is selected on a menu. (Optional) (Does not apply to reactive tasks and background processes.)
Menu Icon	The icon to display next to the name on a menu. See "36.2.2 Specifying a Menu Icon" on page 753 for details. (Optional) (Does not apply to reactive tasks and background processes.)
Parameters	The input parameters that the task or set requires. See "36.2.6 Specifying the Parameters for a Script" on page 755 for details. (Optional) (Does not apply to background processes.)
Command Level	For tasks and sets, you can specify at what level it displays in the Command Panel. Primary is for the most common ones, Secondary is for advanced, and Tertiary is for the least common. See "29.1.1 The Command Panel" on page 558 to learn how these options are used.
Magic Command	For tasks and sets, you can specify whether it is a magic command. Magic commands are options that are not available in reality, such as the Invulnerable and Undetectable set data requests.
Show on Top Level Right Click Menu	Select to show the task or set on the top level of the shortcut menu. This setting can be in addition to showing the script on the Task or Set menu.
One Click Support	Select to allow the task or set to be assigned with a single click. You must assign the keyboard or mouse controls in the Key Mapping Editor as described in "Chapter 86. Creating and Editing Key Mappings" on page 1649.
Show Task on Task Menu	Select to show the scripted task on the Task menu. Clear to hide the task. (You might want to write a script that will be used by other scripts, but which you do not want a user to assign directly to a simulation object.) (For task scripts only.)
Show Set on Set Menu	Select to show the scripted set data request on the Set menu. Clear to hide the set. (You might want to write a script that will be used by other scripts, but which you do not want a user to assign directly to a simulation object.) (For set scripts only.)
Menu Location	The location on the Task or Set menu where this script will be listed. See "36.2.7 Specifying a Menu Location" on page 764 for details. (Does not apply to reactive tasks and background processes.)

Table 34. Script metadata (continued)

Input Field	Description
Show Task on Task Toolbar	Select to add this task to the Task Toolbar. The toolbar displays the name of the script. If you want to display an icon, select a Task menu icon for this task, as described in "36.2.2 Specifying a Menu Icon" on the facing page. (Tasks only.)
Show Set on Set Toolbar	Select to add this set data request to the Set Toolbar. The toolbar displays the name of the script. If you want to display an icon, select a menu icon for this task, as described in "36.2.2 Specifying a Menu Icon" on the facing page. (Sets only.)
Make Available Based on Entity Type	If selected, the script is available to the simulation object types in the Valid for Entity Types list.
Valid for Object Types	The simulation object types that can use this script. Editable only if Make Available Based on Entity Type is selected. This entry affects the context sensitivity of the Task or Set menu. See "36.2.9 Configuring a Script's Availability to a Simulation Object Type" on page 766 for details. Default: All.
Make Available Based on Behavior Sets	If selected, the script is available to the simulation object only if the selected behavior set is assigned to its force.
Behavior Sets	Select each behavior set that you want this script to be part of. You can also assign scripts to behavior sets in the Edit Behavior Sets dialog box. For details about behavior sets, see "29.6. Using Behavior Sets to Manage Scripts" on page 577.
Make Available to Global Plans	If selected, the script is available on the Task or Set menu in global plans.
Toolbar Location	The location on a toolbar for this script. You can assign more than one script to a toolbar button. For details, see "36.2.10 Specifying Equivalent Scripts for a Toolbar Button" on page 769.
Enabled by Default	Reactive tasks only. If selected, the reactive task is automatically enabled for all supported simulation object types.
Default Priority	Reactive tasks only. The priority for this task. For information about the effect of reactive task priority, see "29.5. Reactive Tasks" on page 572.
Compare Priorities	Reactive tasks only. Displays a list of the reactive tasks that are valid for this task's simulation object types and action categories. This helps you decide what priority you want to give to the reactive task you are creating or editing.

36.2.1 Specifying the Script ID

Each script must have a unique ID. VR-Forces generates a default ID, but it does not contain meaningful text. Therefore, you may want to change the ID.

> ⚠ The script ID is used internally by VR-Forces to manage how a script is executed. If a script is used in a plan, it is referenced by its script ID. If you change the ID, you must ensure that it is unique.

➤ **To specify the script ID:**

1. In the New Script (or Edit Script) dialog box, click the button to the right of the Script ID. The New Script ID dialog box opens.
2. Type the ID you want to use. Internally, VR-Forces converts all script IDs to lower case. Therefore it is recommended that you use lower case when creating a script ID.
3. Click OK.

Editing a Script ID

Editing script IDs is not recommended because it can invalidate scripts that reference the changed script. However, you can change a script ID if you want to.

If you change the script ID for a scenario script, you are responsible for ensuring that scripts that reference it are changed accordingly.

If you change the script ID for a system script, VR-Forces generates an alias so that scripts that reference the old script ID will continue to work.

➤ **To edit a script ID:**

1. Open the script whose ID you want to change.
2. Click the button to the right of the Script ID. The New Script ID dialog box opens.
3. Type the ID you want to use. Internally, VR-Forces converts all script IDs to lower case. Therefore it is recommended that you use lower case when creating a script ID.
4. Click OK.

36.2.2 Specifying a Menu Icon

If you want to put an icon on the Task or Set menu next to the menu text, you can specify one. (Does not apply to reactive tasks and background tasks.)

➤ **To specify a menu icon:**

1. Click the Add button () next to the Menu Icon label. The Select Menu Icon dialog box opens.
2. Browse the directory structure and select the icon that you want to use.
3. Click Open.

➤ To remove a menu icon for a script, click the Remove button (🗑) next to the Menu Icon label.

36.2.3 Specifying an Extended Name

Sometimes a script needs to be implemented differently for different types of simulation objects. To accomplish this you can write different scripts for each type of simulation object, but give the task the same name. The scripts are distinguished by their script IDs and VR-Forces picks the correct script using context sensitivity. For example, you might have two scripts with the IDs fly_loiter_pattern_fwa and fly_loiter_pattern_rwa. Both would have the name Fly Loiter Pattern. When you wanted to assign that task, you would select the simulation object and choose Fly Loiter Pattern from the Task menu or command panel. VR-Forces would pick the correct version of the task.

The one problem with this system is that in cases where VR-Forces does not enforce context-sensitivity of task or set lists, you would end up with two tasks called Move to Waypoint and no way to know which was the correct one to use for the simulation object that you wanted to assign the task to. The extended name solves this problem. If you specify an extended name for a script, it is used to distinguish between scripts that use the same task or set name.

36.2.4 Specifying a Short Description

When you add a scripted task or set data request to a plan, the entry in the Plan window can be lengthy and cryptic. It often includes information, such as location values that are not easily understood or put to use by a reader. The Short Description field for a script lets you specify a short, meaningful description to be used in the Plan window.

The description can include variable information. To include variable data, use the syntax `$variable_name`, where *variable_name* is one of the variables for the script. For example, the default listing for the Move To task is:

```
move-to: destination=Waypoint1
```

If you specify a short description of Move to `$destination`, the listing in the Plan window would be:

```
Move to Waypoint1
```

In this case, the value of the destination parameter replaces the text `$destination` in the description.

36.2.5 Specifying a Description

You can include a long description of the script. You can do this solely for the benefit of someone examining the metadata in the Edit Script dialog box or you can include the text of the description in the dialog box that is generated for the scripted task or set. This is a way of providing context-sensitive help to the person who must specify the parameter values for the task or set.

➤ **To specify a task or set description:**

1. In the Description box, type a description of the task or set.
2. If you want the description to appear on the generated dialog box, select the Description check box.

36.2.6 Specifying the Parameters for a Script

Scripted tasks and sets can take input parameters. When you specify parameters for a script, VR-Forces automatically builds a dialog box where you can enter the parameter values.

Script Parameter Types

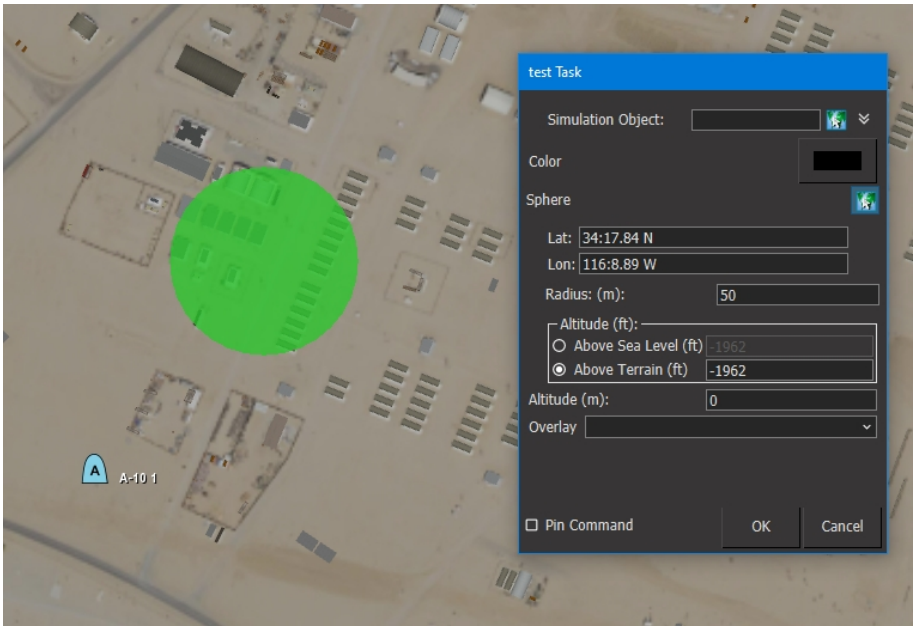
The parameter types that you can specify for a script are:

- Active Sonar Mode. Allows you to select the sonar mode to use.
- Aerodrome. Allows you to select an aerodrome.
- Aggregate Formation. Provides a list of formations for units.
- Air Speed. Allows you to specify the speed for an aircraft.
- Altitude. Allows you to specify the altitude.
- Altitude Label. Adds an altitude label independently of the altitude parameter.
- Altitude Rate Change. Allows you to specify the rate at which an aircraft changes altitude.
- Angle. Allows you to specify an angle.
- Animated Movement. Lets you assign an animated movement to an entity.
- Bomb Resource. Creates a list of bomb resources.
- Boolean. Creates a check box.
- Chat Message Channel.
- Choice (Drop-Down List). Specifies choices presented in a drop-down list.
- Choice (Image Drop-Down List). Lets you specify a set of images to choose instead of text.
- Choice (Multiple Option Buttons). Specifies choices presented as a group of option (radio) buttons.
- Choice (Single Option Button). Specifies a choice presented as a single option (radio) buttons. Single option buttons can be associated through a button group, but placed at different locations on the dialog box.
- Color. Allows you to select a color.
- Country. Allows you to select a country.
- DI-Guy Animation. Allows you to select a DI-Guy animation.
- DI-Guy Appearance. Allows you to select a DI-Guy appearance.
- DI-Guy Gesture. Allows you to select a DI-Guy gesture.

- Date/Time. Specifies the date and time.
- Depth MSL. Specifies the depth below mean sea level.
- Direct Fire Weapon. Lets you select a direct fire weapon.
- Distance (Large). Specifies a large distance. This uses the Application Settings level's Distance measurement from the Display Units Settings.
- Distance (Small). Specifies a small distance. This uses the Application Settings level's Distance (Small) measurement from the Display Units Settings.
- Embarkation Slot. Allows you to select a slot an entity can embark on.
- Embedded Entity. Allows you to select an embedded entity to deploy.
- Filename. Allows you to specify a file name.
- Force. Specifies the force.
- Heading. Heading, in degrees.
- Heading Label. Adds a heading label independently of the heading parameter.
- Hex. Allows you to enter a hexadecimal.
- Indirect Fire Weapon. Specifies an indirect fire weapon.
- Integer. A whole number.
- Integer Map. Maps a keyword to an integer.
- Jammer Selector. Allows you to select a jammer for an entity. The jammer types list is built automatically based on the entity type.
- Jamming Mode. Lets you select a jamming mode. The jamming mode list is built automatically based on the entity type. If you add a Jamming Mode parameter, VR-Forces automatically adds an Emitter ID parameter to the dialog box.
- Loadout. Specify a loadout for an aircraft unit. (Aggregate-level only.)
- Location (with altitude). Vector description of a location, in geocentric, including altitude.
- Location (without altitude). Vector description of a location, in geocentric, altitude not included.
- Location Reference (with altitude). Allows you to specify a location using either coordinates or a waypoint, including altitude.
- Location Reference (without altitude). Allows you to specify a location using either coordinates or a waypoint, altitude not included.
- Munition Resource. Creates a list of munitions.
- Name Tag. Specifies a name tag for a DI-Guy soldier.
- Naval Mine Resource. Specifies a naval mine resource.
- New Tactical Graphic. Allows you to define a tactical graphic.
- Object Type. Allows you to select one type of object.
- Object Types. Allows you to select multiple object types.

- Observer Mode. Specifies an observer mode, for example, Stealth, Plan View, or XR.
- Observer Saved View. Specifies a saved view.
- Octal. Allows you to enter an octal code.
- Offset Location. Specifies an offset to the side, behind, and above, as in the Follow Entity task. The default is x=10;y=20;z=30.
- Overlay Parent. Allows you to select an overlay.
- Pitch. Allows you to specify the pitch for a simulation object.
- Radar Mode. Allows you to specify a radar mode. The radar mode list is built automatically based on the entity type. If you add a Radar Mode parameter, VR-Forces automatically adds an Emitter ID parameter to the dialog box.
- Radar Selector. Allows you to select a type of radar. The radar list is built automatically based on the entity type.
- Range Label. Adds a range label.
- Range Label (Small).
- Rank Patch. Specifies a rank patch for a DI-Guy soldier.
- Rate Label. Adds a rate label.
- Real. A floating point decimal number.
- Real Map. Maps a keyword to a real number.
- Resource. Creates a list of available resources.
- Runway Selector. Allows you to select an aerodrome's runway.
- SISO Pick Widget. Specifies a SISO document.
- Separator. Allows you to put a separator in the generated dialog box. This is just for formatting purposes.
- Simulation Object (Single). Allows you to select one simulation object in a list window.
- Simulation Objects (Multiple). Allows you to select multiple simulation objects in a list window.
- Speed. Any kind of rate, such as speed, climb rate, and so on.
- Sphere. Allows you to create a sphere for defining an area on the terrain (Figure 258).

Figure 258. A sphere created as part of a scripted task



- String. An alphanumeric string.
- String Map. Maps a keyword to a string.
- Subordinate Functions. Assign a subordinate function.
- Subordinates. Allows you to select one or more of a unit's subordinates.
- Time. Time in days:hours:minutes:seconds.
- Topographic Orientation. Specifies orientation in topographic coordinates.
- Turn Rate. Turn rate in radians/second.
- Turn Rate Label. Adds a turn rate label independently of the turn rate parameter.
- Unit Patches. Specifies a unit patch for a DI-Guy soldier.
- Visual Meta Data. Allows you to select markings for objects that support settings marking, such as a flag or license plate.
- Weapon. Specifies a weapon.

Specifying Script Parameters

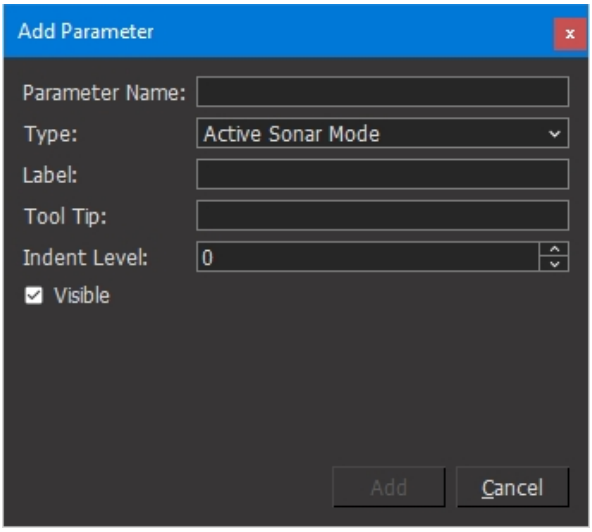
This section explains how to add parameters using the VR-Forces script creation interface and automatic dialog box creation. You can also add parameters using Qt Designer. For details, see "39.3. Adding Parameters When Using Qt Designer" on page 854.

If a script does not have any input parameters, it executes as soon as it is assigned to a simulation object, like the Wait command does.

➤ **To specify a parameter for a script:**

1. Click the Add button ( ) above the Parameters list. The Add Parameter dialog box opens (Figure 259).

Figure 259. Add Parameter dialog box



2. In the Parameter Name box specify a name for the parameter. This is the name used in scripts to get the parameter value.
3. Select the type of parameter from the Type list. If appropriate, the dialog box redisplays to request the details required for the parameter type you selected.
4. Specify the details for the parameter. Table 35 explains how to fill out the properties for the various parameter types.
5. Click Add.

Table 35. Scripted task parameter properties

Parameter property	Description
Parameter Name	The name to be used by the Lua script for this parameter. For example, if you give a parameter the name **taskDestination**, in the Lua code for the script, the parameter will be accessed as **taskParameters.taskDestination**.
Label	This is the text used on the dialog box that is generated for the task. If you leave this field blank, the parameter name is used.
Button Group	For Choice (Single Option Button) parameters, specifies the button group that the option buttons are a part of.
Tool Tip	Text for a tool tip. (Optional)

Table 35. Scripted task parameter properties (continued)

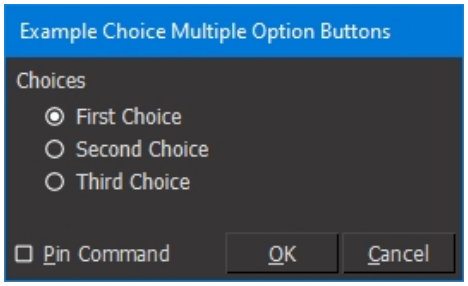
Parameter property	Description
Indent Level	The level of indentation for this parameter. The indentation level is multiplied by the Indent setting (in pixels) on the Application Settings dialog box, Script Settings page. This allows you to apply some formatting to the dialog box. For details about setting the indentation distance, see "Setting the Indentation Distance for Script Dialog Boxes" on page 763.
Required	For Simulation Object lists only, specifies that this parameter must be set before the OK button is enabled. If the check box is cleared, the requirement is not in effect.
Visible check box	Specifies that the parameter be included on the script dialog box. (You might want a parameter to be available in the code, but not have it set by a user when the script is assigned.)
Default value or state	Many, but not all, parameters have an optional default value.
Initialize from State Property	This is available for some parameters. Select the check box if you want to indicate that it is associated with a state property. If you do, you can enter the state property name. In the scripted task dialog, users will see the values for that state properties.
Range Bottom	The lowest acceptable value for this parameter, if applicable.
Range Top	The highest acceptable value for this parameter, if applicable.
Default value for boolean parameter	The Boolean parameter type lets you create a check box. You can specify that the default be selected or cleared.
Affects Controls	If selected, some fields are populated with the actual simulation object state values. For details about the Affects Controls option for the simulation object parameter, see "Using the Affects Controls Option" on page 763.
Choice	Adds choice option buttons. For details about adding choices to a Choice parameter, see "Specifying Choices for the Choice Parameters" on page 762.
Filter category	Lets you add categories for filtering object lists. For details about adding categories to a filtered list, see "Specifying Categories for a Simulation Object" on page 762.

Choosing an Option Button Type

Scripted tasks can use two types of option buttons – multiple or single. The multiple option button choice is a typical option list, as illustrated in (Figure 260). In a multiple option list, the choices are mutually exclusive and are automatically associated with each

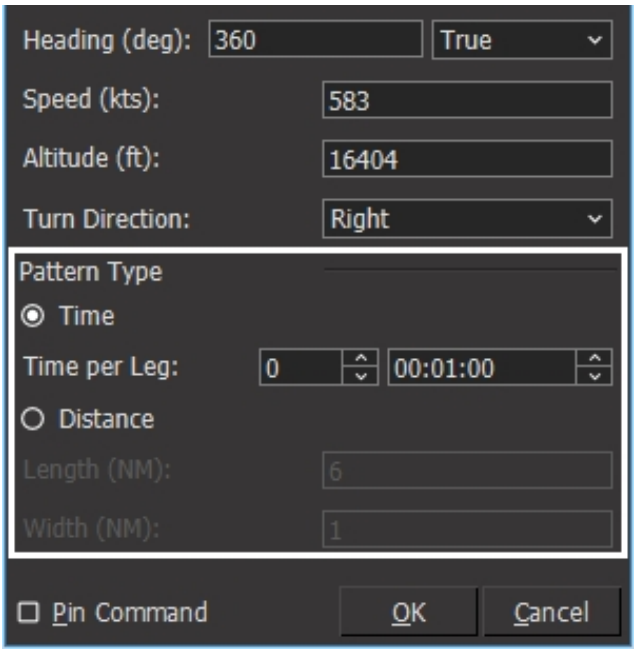
other. You cannot enable or disable other input parameters based on the selection of one of these options.

Figure 260. Choice (Multiple Option Buttons)



A single button option puts one option button on the dialog box. You can create multiple single button options and place them on the dialog box wherever you want. This allows you to associate other parameter inputs with the option button using the dialog box layout (Figure 261). To ensure that these options are mutually exclusive, you associate them by specifying a button group.

Figure 261. Choice (Single Option Button)



You can create multiple sets of options and multiple buttons groups if you need to.

Although you can lay out single option buttons and associated parameters using the default GUI builder for scripts, to associate parameters with the options, you must use Qt Designer. For details about laying out dialog boxes using Qt Designer, see "Chapter 39. Using Qt Designer for Script Dialog Boxes" on page 853.

Specifying Choices for the Choice Parameters

The Choice (Drop-Down List), Choice (Image Drop-Down List), and Choice (Multiple Option Buttons) parameters require a list of choices, which are added to the dialog box as option buttons or as a list. For the image drop down list, you specify images. For the other two, you specify text. You can add choices, edit them, and delete them.

➤ **To add a text choice:**

1. Click the Add Parameter button ( ) on the Choice Box Values line. The New Choice dialog box opens.
2. Type the text for the choice.
3. Click OK.
4. Repeat this procedure for each choice that you want to have available. (The last choice added becomes the default choice.)

➤ **To add an image choice:**

1. Click the Add Parameter button ( ) on the Choice Box Images line. The Select Image dialog box opens.
2. Type the path for an image or click the Edit button and select an image in the file browser. The image displays in the dialog box.
3. Click OK. The image is added to the list next to Choice Box Images.
4. Repeat this procedure for each image that you want to have available. (The last choice added becomes the default choice.)

Specifying Categories for a Simulation Object

The Simulation Object parameter lets you build a list of categories on which to filter the display of simulation object types in a dialog box.

➤ **To add a category to the simulation object list:**

1. Click the Add Parameter button ( ) on the Filters line. The Choose Filter dialog box opens.
2. Select a filter from the Filters list.
3. Click OK.
4. Repeat the procedure to add as many filters as you want.

Editing the Category List for a Simulation Object Parameter

You can edit the list of filter categories for a Simulation Object parameter. The edit operation is essentially a replacement of the current category with a new one. You can achieve the same effect by deleting the current category and adding a new one.

➤ **To edit the list of filter categories in a Simulation Object parameter:**

1. Select the parameter you want to edit in the Parameters list on the New Scripted Task dialog box.

2. Click the Edit button ( ). The Script Variable dialog box opens.
3. In the Filters list, select the filter that you want to edit.
4. Click the Edit button ( ). The Choose Filter dialog box opens.
5. Select the filter category that you want to use.
6. Click OK. The original category is removed and the new one is added.

➤ **To delete a category from a Simulation Object parameter:**

1. Select the parameter you want to edit in the Parameters list on the New Scripted Task dialog box.
2. Click the Edit button ( ). The Script Variable dialog box opens.
3. In the Filters list, select the filter that you want to delete.
4. Click the Delete button ( ). The category is removed from the list.

Using the Affects Controls Option

The Simulation Object (Single) and Simulation Objects (Multiple) parameters have a check box labeled Affects Controls. If you select this check box, when you select a simulation object in the task dialog box, the values for the following parameter types update to the correct values for the selected simulation object:

- DI-Guy Animation
- DI-Guy Appearance
- Embedded Entity
- Emitter
- Force
- Formation
- Location
- Overlay
- Weapon

If you use the Simulation Objects (Multiple) parameter, it only affects controls if you select just one simulation object.

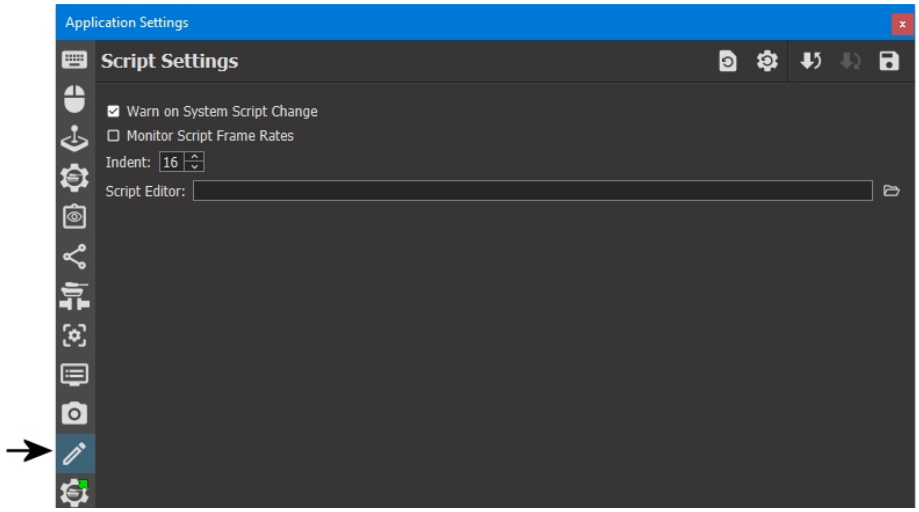
Setting the Indentation Distance for Script Dialog Boxes

When you add parameters to a script dialog box, you can specify an indent level. The indent level is the level of indentation, not the distance of indentation. The distance that the parameter is indented on the dialog box is the indent level multiplied by the Indent value (in the Application Settings dialog box, Script Settings page.

➤ **To set the indentation distance for script dialog boxes:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 262).

Figure 262. Script Settings page



3. Change the value (in pixels) in the Indent box.
4. Click Close.

You can also open the Script Settings page from the Scripts menu in the Scripts dialog box.

36.2.7 Specifying a Menu Location

Scripted tasks are added to the Task menu unless you specify that they not be shown (by clearing the Show Task on Task Menu check box). Scripted set data requests are added to the Set menu. The default location is the top level of the menu. However, you will probably want to organize your tasks and sets using the submenus provided by VR-Forces or in newly created submenus. The New Script dialog box displays the menu under which the script is placed. It does not show the full path from the top level menu.

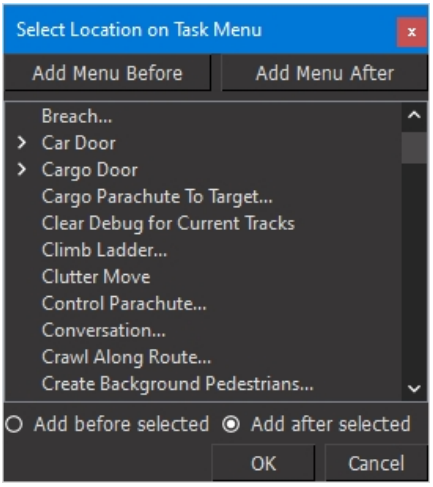
You can also add a scripted task or set to the top level shortcut menu. The script can be listed in both places.

- When you add a script to a menu, it does not appear on the menu until you reload the scenario.
- By default, VR-Forces alphabetizes all Task and Set menu commands. Therefore even if you specify a specific location on a menu, the task or set will be listed in alphabetical order unless you disable alphabetization. For information about alphabetizing Task and Set menu commands, see "85.2.1 Alphabetizing the Task and Set Menus" on page 1640.

➤ **To specify a menu location for a script:**

1. On the Menu Location line, click the Edit button (). For task scripts, the Select Location on Task Menu dialog box opens (Figure 256 on page 748). For set scripts, the Select Location on Set Menu dialog box opens. The two dialog boxes are functionally identical.

Figure 263. Select Location on Task Menu dialog box



2. To add the script to an existing submenu:
   a. Expand the submenu on which you want to put the script.
   b. Select the menu item that you want the new script to be next to.
   c. To put the new script above the selected menu item, select the Add Before Selected option. To put the new script below the selected menu item, select the Add After Selected option.
   d. Click OK.
3. To create a new submenu for the script:
   a. Select the submenu next to where you want to add the new submenu.
   b. To add the new menu before the selected submenu, click Add Menu Before. To add the new menu after the selected submenu, click Add Menu After. The New Menu Name dialog box opens.
   c. Type a name for the new submenu.
   d. Click OK.
   e. Add the script to the new submenu using the procedure in step 2.

36.2.8 Specifying Action Categories

Some VR-Forces tasks can run at the same time as other tasks, while some are mutually exclusive. This is called task concurrency. (For details about concurrent tasks, see "29.1.4 Concurrent Task Execution" on page 561.) The C++ tasks provided with VR-Forces are organized in groups to help manage concurrency. They are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/default-task-rules.tsk*.

In the New Script dialog box, these groups are represented as action categories. The action categories displayed in the New Script dialog box are configured in *./data/simulationModelSets/<model_set>/vrfSim/taskRules/actionCategories.tsk*. (For details about configuring action categories, see "29.7.1 Configuring Action Categories" on page 581.)

When you create or edit a scripted task, you can specify which groups of tasks the new task can run concurrently with and which it will interrupt. By default, the Movement category is selected, which means that if you do not change the setting, the new task will interrupt movement tasks.

➤ To specify which tasks a scripted task will interrupt, in the Action Categories list, select the categories to interrupt. For example, if the new task involves movement, you probably want it to interrupt other movement tasks, so you would select the Movement category check box. If it is a new type of message task, you probably want to let it run concurrently with movement tasks, so you would clear the Movement category check box.

36.2.9 Configuring a Script's Availability to a Simulation Object Type

Scripts can be available to a simulation object based on these criteria:

- The script is available for certain simulation object types. For details, see "Specifying the Valid Simulation Object Types for a Script" on the facing page.
- The script is part of a behavior set. For details, see "Specifying Availability by Behavior Set" on the facing page.
- A task or set is specified in a system definition. (For example, the Naval Mine Sweep task is available to entities with a Naval Mine Sweep system.) This state is not shown in the Edit Script dialog box. For details, see "Specifying a Scripted Task or Set in a System Definition" on page 768.

The default setting for new scripts is that they are available to all simulation objects.

Availability by simulation object type and by behavior set is configured in the New Script and Edit Script dialog boxes. Table 36 shows the interaction of these settings.

Table 36. Scripted task availability

Available by Entity Type	Available by Behavior Set	Availability
☑	☐	The script is available only to those simulation object types listed.
☐	☑	The script is available to all simulation objects, but only if one of the selected behavior sets is applied to the simulation object's force.
☑	☑	The script is available to the specified simulation object types, but only if one of the selected behavior sets is applied to the simulation object's force.

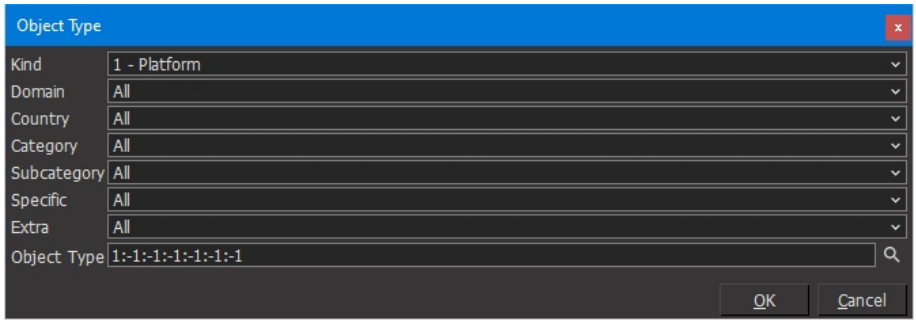
Specifying the Valid Simulation Object Types for a Script

You can restrict a script to specific simulation object types. If you do this, the script will not appear on the Task or Set menu when you select a simulation object for which it is not valid. Just as with built-in VR-Forces tasks and sets, there are cases, such as for global plans, where VR-Forces cannot determine the simulation object context and the script will be available on the menu even though it is not appropriate to the simulation object that you selected. In those cases it is up to you to know that a task or set is valid.

➤ **To specify the valid entity types for a script:**

1. Select the Make Available Based on Entity Type check box.
2. Click the Add button ( ) on the Valid for Entity Types line. The Object Type dialog box opens (Figure 264). You can build an enumeration by selecting options from the enumeration field lists or you can type in the enumeration directly.

Figure 264. Object Type dialog box



3. For each field of the simulation object type (Kind, Domain, Country, and so on), select an option from the list.

   The options are drawn from the SISO Enumerations document. As you select an option, the enumeration at the bottom of the dialog box changes to reflect your choice. You do not need to make a selection for each field because VR-Forces can use wild carding to match simulation objects against the final enumeration.
4. If appropriate, add additional enumerations for each simulation object type that is valid for the script.

Specifying Availability by Behavior Set

You can restrict a script to specific behavior sets. If you do this, the task will not appear on a menu when you select a simulation object whose force does not have the behavior set assigned to it.

➤ **To specify that a script is available by behavior set:**

1. Select the Make Available Based on Behavior Set check box.
2. Select each behavior set for which you want this script to be available.

ⓘ You must create the behavior sets before you can assign scripts to them. All available behavior sets are automatically listed in the dialog box window.

Specifying a Scripted Task or Set in a System Definition

Some scripted tasks and sets are available to simulation objects because they are specified in a system that the simulation object uses. They are not assigned to simulation object types as part of the script metadata.

> This method is used for many of the system scripted tasks supplied with VR-Forces.

Scripted tasks and sets are specified in a system definition by their script IDs in the **script-ids** parameter. (In aggregate-level SMSs, some scripted tasks for combat engineering objects are specified in the **create-script-id**, **improve-script-id**, and **destroy-script-id** parameters.) Script IDs can be specified in a script-enable-controller and in controllers derived from it.

The following example is from *mech-aggregate-movement.sysdef* in *AggregateLevelBase.sms*. Any simulation object with this system definition can use the Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_Plan_Path, Move_To_Waypoint_Plan_Path, and other scripted tasks. The variables described in the script-variables block are accessible to the script through the vrf:getScriptAttribute() function.

```
(script-enable-controller
 (component-descriptor-type "script-enable-controller-descriptor")
 ...
 (script-ids "Maximum_Speed_Modifier" "Halt_Movement"
      "Move_To_Location_Plan_Path" "Move_To_Waypoint_Plan_Path"
      "halt_movement_before_obstacles" "Flee_When_Routed"
      "React_to_indirect_fire")
 (script-variables
    (maximum-speed-modifier-variables
       (script-id "Maximum_Speed_Modifier")
        (variables
          (DtRwReal Rain-Modifier-By-Intensity
             $rain-modifier-by-intensity)
          (DtRwReal Snow-Modifier-By-Intensity
             $snow-modifier-by-intensity)
          (DtRwReal Visibility-Degrades-When-Below
             $visibility-degrades-when-below)
          (DtRwReal Visibility-Can-Degrade-Speed-Up-To
             $visibility-can-degrade-speed-up-to)
        )
    )
    (move-to-location-variables
       (script-id "Move_To_Location_Plan_Path")
       (variables
          (DtRwString Obstacle-Feature-Query "MAK_OBSTACLE")
          (DtRwString Path-Feature-Query "MAK_ROAD")
       )
    )
    ...
 )
)
```

The following example is from the *aggregate-5-inch-gun.sysdef* in *AggregateBaseLevel.sms*. It adds the **script-ids** parameter to the **aggregate-indirect-fire-controller**, which is derived from the **script-enable-controller**. Any simulation object with this system can use the Indirect Fire and Limited Munition Attack scripted tasks.

```
(weapon-indirect-fire
 (systems )
 (sensors )
 (controllers
    (aggregate-indirect-fire-controller
       (component-descriptor-type "aggregate-indirect-fire-
          controller-descriptor")
       (component-type  "aggregate-indirect-fire-controller")
       (min-tick-period  -1.000000)
       (min-tick-period-variance  -1.000000)
       (is-enabled True)
       (script-ids "Indirect_Fire" "Limited_Munition_Attack")
       (weapon-name  "5 inch shell")
       ...
    )
 )
```

The following example is from *obstacle-builder.sysdef* and shows the **create-script-id** and **improve-script-id** parameters:

```
(creatable-objects
    (affectable-object
       (object-type 1 (16 1 0 62 3 0 0))
       (time-to-complete $time-to-create-abatis)
       (range $creation-range-abatis)
       (create-script-id "Construct_Abatis")
       (improve-script-id "Improve_Obstacle")
    )
    ...
)
```

36.2.10 Specifying Equivalent Scripts for a Toolbar Button

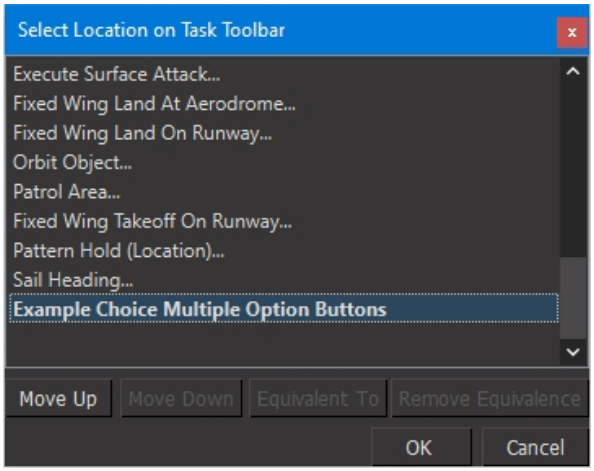
There may be occasions when you want to have tasks or sets with the same name that work differently depending on the simulation object you give them to. For example, you might have a movement task that works one way for an individual entity and another way for a unit. You can ensure that when you select a simulation object it gets the correct version of the script by specifying the simulation object type that the script is available to. To simplify the Task or Set toolbars, you can map these different scripts to the same toolbar button. This is done by specifying that one script is equivalent to another script.

If you map multiple scripts to the same toolbar button, when you select a simulation object, the toolbar button is enabled and you can give the simulation object the task or set. VR-Forces chooses the correct version of the script. If you select multiple simulation objects that would have different versions of the script, the toolbar button becomes disabled.

➤ **To specify an equivalent script for a Task or Set Toolbar button:**

1. Create a new script.
2. On the New Script dialog box (or Edit Script dialog box), select Show Task on Task Toolbar (or Show Set on Set Toolbar).
3. Click the Toolbar Location edit button ( ). The Select Location on Task (Set) Toolbar dialog box opens.
4. Select the task or set that you want this script to be equivalent to. The Equivalent To button becomes active.
5. Click the Equivalent To button. Text is added to the script's entry in the list indicating the script it is equivalent to (Figure 265).

Figure 265. Select Location on Task Toolbar dialog box



Do not move the new script to a different location on the toolbar. Moving it breaks the equivalency.

6. Click OK.
7. Save the script.

36.2.11 Specifying the Programming Language for a Script

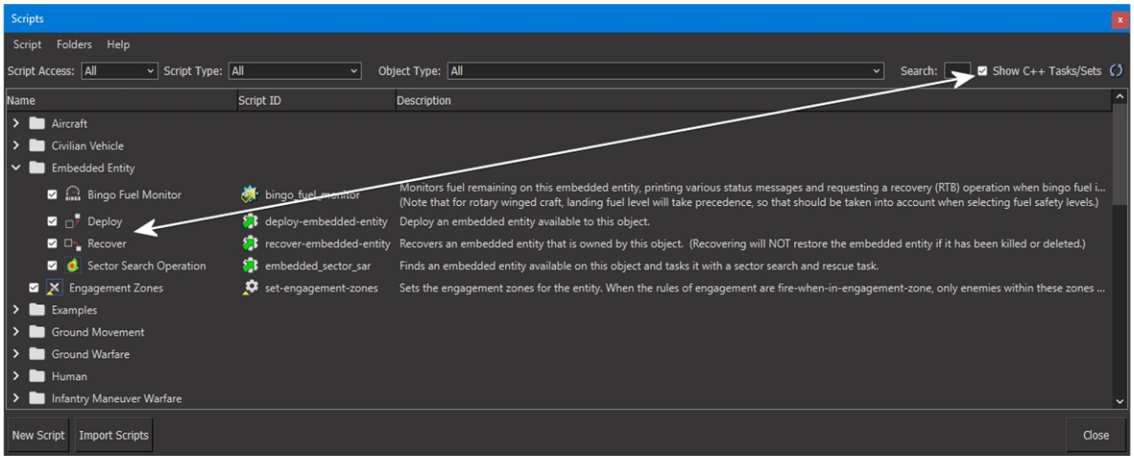
The Language list in the New Script dialog box has two options, Lua and C++. In most cases you will choose Lua and write a script to implement the task or set. However, since the script interface automatically generates dialog boxes and updates menus, it can be a convenient way to add the GUI component for new tasks or sets that developers have written in C++ with the VR-Forces Toolkit. Developers who take this approach do not have to rebuild the VR-Forces GUI or write a plug-in to add their new task or set to the menu and code a dialog box.

The Language option only applies to Task and Set type scripted tasks. Reactive tasks and background processes are always written in Lua.

If a developer adds a new C++ task controller using this approach, it looks for the script ID of the scripted task. For example, the Deploy and Recover tasks for embedded entities use the scripted task interface to create their dialog boxes. The addTask example in the VR-Forces Toolkit demonstrates this method of adding tasks. For more information, please see the *VR-Forces Developers Guide*.

You can display C++ tasks and sets in the Scripts dialog box by selecting the Show C++ Tasks check box (Figure 266).

Figure 266. C++ tasks



36.2.12 Configuring Scripted Tasks to Run Concurrently

You can specify that an entity be able to run more than one instance of a task at the same time. The concurrent task feature is intended for tasks that require an entity to manage two non-conflicting activities, such as when an air base launches a flight in an aggregate-level scenario. If an air base receives two launch flight tasks in close succession and concurrent tasks are allowed, both tasks would complete. If this option is not in effect, the second task would cause the air base to stop executing the first instance of the task.

> This setting overrides the action category settings. It is your responsibility to make sure that you do not allow concurrent task execution for tasks that would not make sense to run concurrently.

➤ To specify that a task can be run concurrently, select the Can Be Executed More Than Once at a Time check box on the Edit Scripts dialog box.

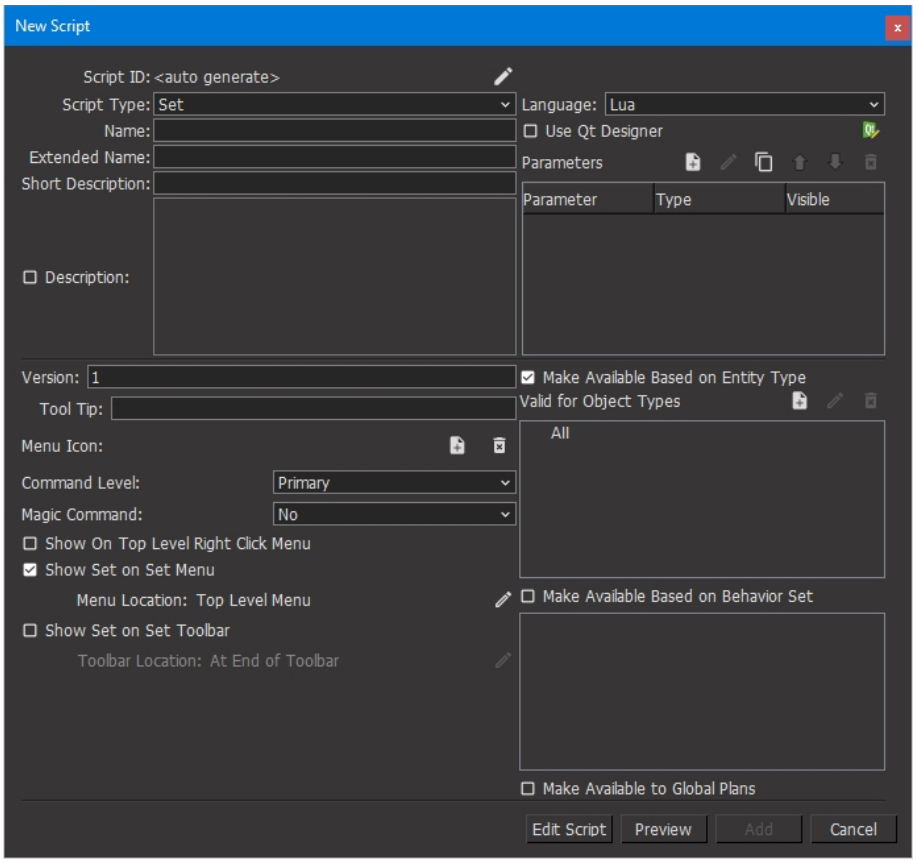
36.2.13 Creating Scripted Sets

The metadata for a scripted set is identical to that of a scripted task except that a scripted set does not have action categories.

➤ **To create a scripted set:**

1. Choose **Simulation > Scripts**. The Scripts dialog box opens (Figure 256 on page 748).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, see "36.6. Organizing Scripts into Folders" on page 777.)
3. Choose **Script > New Script** or click the New Script button. The New Script dialog box opens (Figure 257 on page 749).
4. In the Script Type list, select Set. The New Script dialog box redisplays to show the metadata fields for a scripted set (Figure 267 below).

Figure 267. New Script dialog box, scripted set



5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 34 on page 750.

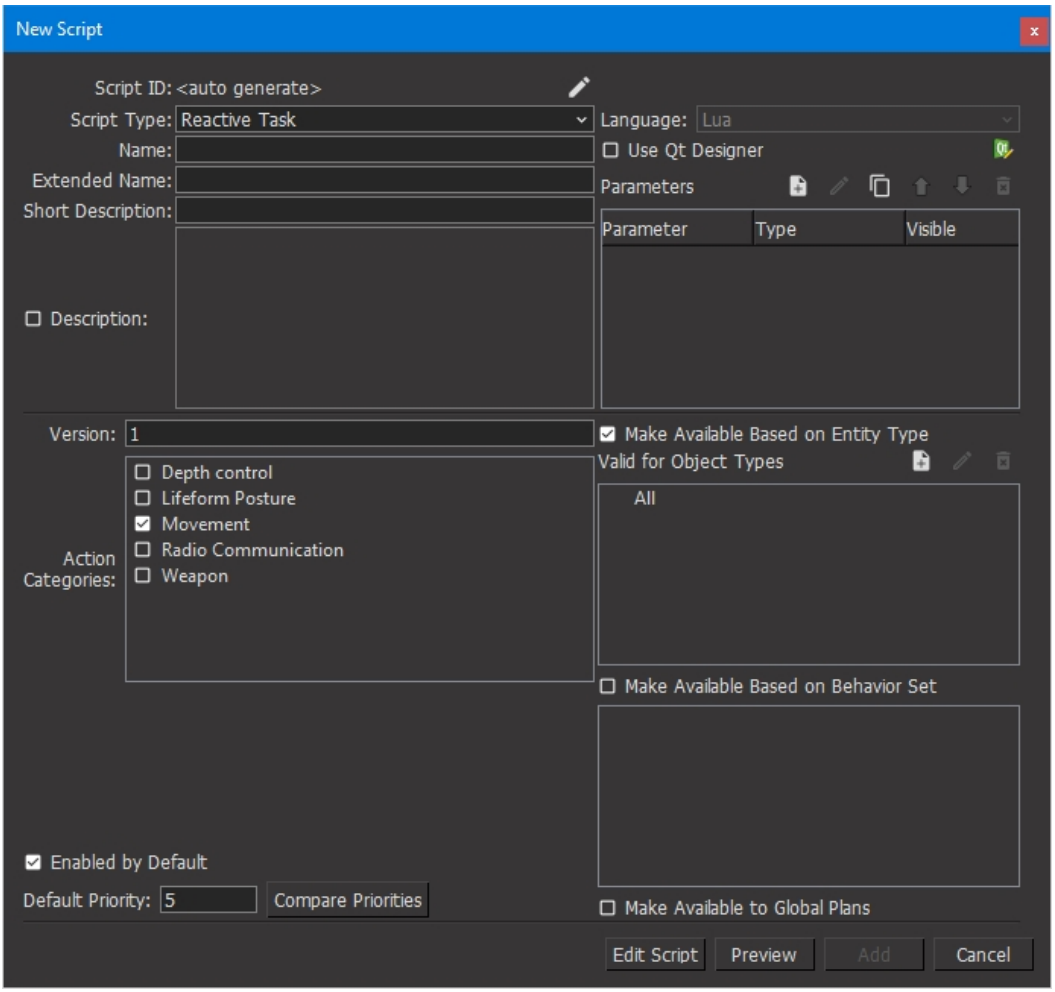
36.2.14 Creating Reactive Tasks

The process for creating a reactive task is largely the same as for creating other scripts. This procedure focuses on the differences.

➤ **To create a reactive task:**

1. Choose **Simulation > Scripts**. The Scripts dialog box opens (Figure 256 on page 748).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, see "36.6. Organizing Scripts into Folders" on page 777.)
3. Choose **Script > New Script** or click the New Script button. The New Script dialog box opens (Figure 257 on page 749).
4. In the Script Type list, select Reactive Task. The New Script dialog box redisplays to show the metadata fields for a reactive task (Figure 268).

Figure 268. New Script dialog box, reactive task



5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 34 on page 750.
6. To make this reactive task enabled by default, select the Enabled By Default check box.

7. In the Default Priority box, type the priority for this task. The lower the number, the higher the priority. To help you set the priority, you can view a list of all reactive tasks that are valid for the simulation object types of this task so that you can consider their relative importance and set an appropriate priority.
8. Optionally, click Compare Priorities to see a list of reactive tasks and their priorities.
9. Click Add.

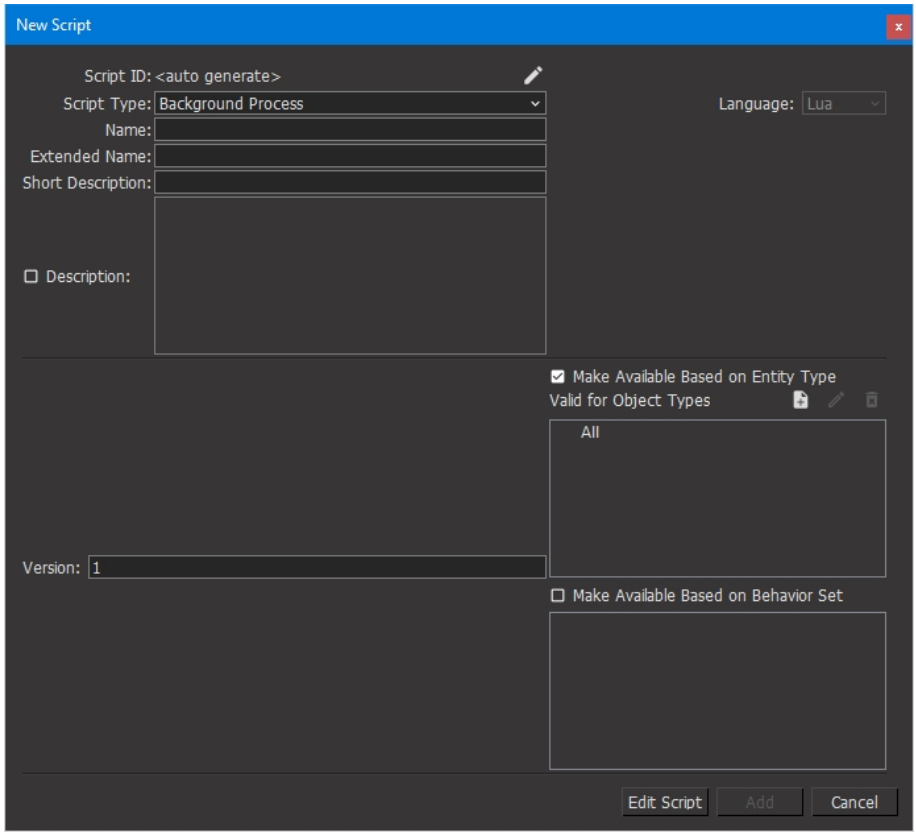
36.2.15 Creating Background Processes

The procedure for creating a background process is essentially the same as for other script types. The main difference is that there are fewer fields to complete in the New Script dialog box. For details about writing a background process script, see "37.11. Background Processes" on page 812.

➤ **To create a background process:**

1. Choose **Simulation > Scripts**. The Scripts dialog box opens (Figure 256 on page 748).
2. If you have a folder hierarchy and want to create the script in a particular folder, select the folder. (For more information, see "36.6. Organizing Scripts into Folders" on page 777.)
3. Choose **Script > New Script** or click the New Script button. The New Script dialog box opens (Figure 269).
4. In the Script Type list, select Background Process. The New Script dialog box redisplays to show the metadata fields for a background process (Figure 257 on page 749).
5. Complete the fields in the dialog box. Fields that are common to all scripts are described in Table 34 on page 750.

Figure 269. New Script dialog box, background process



6. Click Add.

36.3. Saving Scripts

When you create or edit a script, its listing in the Scripts window is in italics, prefaced by an asterisk. When you save the scenario, or promote a script to a system script, the script is saved and the italics and asterisk are removed. If you do not perform one of these actions before closing VR-Forces, your changes will be lost.

It is likely that you will test scripts by running a scenario and you will want to edit the script as you find problems. If you create or edit a script while a scenario is running and you decide to rewind the scenario without saving it (which would be normal behavior for rewinding a scenario), you are prompted to preserve the changes to the script after the scenario rewinds. The script is still not saved and you will, therefore, receive the standard "scenario modified" prompt when you start the scenario again. You do not have to save the scenario if the only thing that changed is the script. You can continue to run the scenario and rewind while you are working on the script. Eventually you must save the scenario if you want to save your changes to the script.

Metadata for system scripts is saved to a file in the format *./data/simulationModelSets/<model_set>/scripts/script_name.xml*. The script name is always saved as lower case.

➤ To save a scenario script, save the scenario.

➤ To save a system script, after updating the script in the Edit Script dialog box, click Update.

36.4. Editing a Script

You can edit scripts.

> ⚠ If you edit a system script, the changes affect all scenarios that use the script. You are responsible for ensuring that any changes have the intended results for all scenarios that are affected.

➤ **To edit the metadata for a scenario script:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the script that you want to edit.

   > ⓘ In the Script ID column, the icon indicates the type of script: Task (), Reactive Task (), Set (), or Background Process ().

3. Choose **Script > Edit Script Meta Data**, or double-click the script name. The Edit Script dialog box opens. (If you are editing a system script, you are prompted to confirm that you want to edit it.)
4. Edit the metadata as desired.
5. Click Update.

36.5. Filtering the List of Scripts

You can filter the tasks listed in the Scripts dialog box by selecting options from the lists below the menu bar:

- Script Access: Show system scripts, scenario scripts, or all scripts.
- Script Type: Show scripted tasks, reactive tasks, sets, background processes, or all scripts.
- Entity Type: Show scripts that apply to the selected simulation object type.

  > ⓘ This filter is based on the Valid for Entity Types list for each script. Some scripts do not have any simulation object types listed. These scripts, such as Drop Naval Depth Charge, become available to simulation objects based on the systems that they have configured. Therefore, this list may be somewhat misleading in terms of the overall applicability of scripts to the different platforms.

- Show C++ Tasks: Toggles the display of tasks and sets that are written in C++, but whose GUIs are created using the script mechanism.

36.6. Organizing Scripts into Folders

You can create folders and organize scripts by folder.

36.6.1 Adding a Folder

➤ **To add a folder:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Choose **Folders > Add Folder**. The Folder Name dialog box opens.
3. Type a name for the folder.
4. Click OK.

36.6.2 Renaming a Folder

➤ **To rename a folder:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the folder that you want to rename.
3. Choose **Folders > Rename Folder**. The Folder Name dialog box opens.
4. Type a new name for the folder.
5. Click OK.

36.6.3 Deleting a Folder

You cannot delete a folder that has scripts in it.

➤ **To delete a folder:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the folder that you want to delete.
3. Choose **Folders > Delete Folder**.

36.6.4 Adding Scripts to a Folder

➤ To add a script to a folder, drag it onto the folder.

36.6.5 Removing a Script from a Folder

➤ To remove a script from a folder, drag it away from the folder.

36.7. Exporting and Importing Scripts

Scenario scripts are saved in a scenario's SPT file, not as individual scripts. If you want to make a scenario script available to other scenarios, but you do not want to save it as a system script, you can export the script from the scenario in which you created it and import it into any other scenario. You can export multiple scripts to one file as a script "package".

> You cannot add or remove scripts from a script package. To change the contents you would have to create a new package.

If scripts are organized in folders, when you save them the folder structure is also saved.

➤ **To export scripts:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the scripts that you want to export. (The window supports typical methods for selecting multiple entries in a list.)
3. Choose **Script > Export Script Package**. The Choose Script File dialog box opens. By default it opens in *./userData/taskScripts*.
4. Type a name for the scripts file.
5. Click Save. The scripts are saved to a file with the extension *.spt*. The file contains the metadata and the script code.

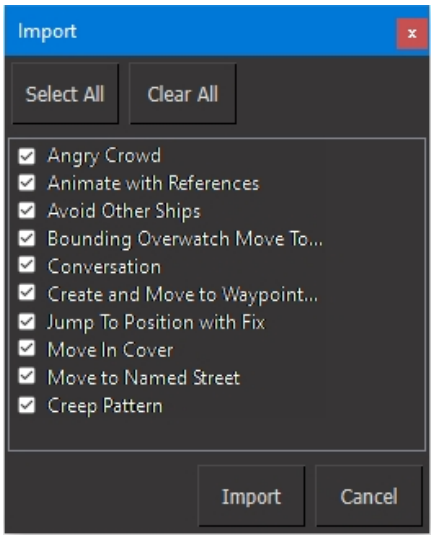
36.7.1 Importing a Script Package

If you import a script package that has multiple scripts, you can choose which scripts to import. Scripts are imported with the folder hierarchy that they were saved in.

➤ **To import scenario scripts:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Choose **Script > Import Script Package** or click Import Scripts. The Choose Script File dialog box opens.
3. Select the script file that you want to import.
4. Click Open. The Import dialog box opens (Figure 270). It lists the scripts saved in the script file.

Figure 270. Import dialog box



5. Select the scripts that you want to import.
6. Click Import.

36.8. Copying a Script

If you want to create a new script that is similar to an existing one, you can copy an existing script and then edit it for its new purpose.

➤ **To copy a script:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the script that you want to copy.
3. Choose **Script > Duplicate**. The New Script dialog box opens. If you copied a scenario script, the duplicate script is assigned a new script ID. If you copied a system script, the script ID is not changed. You will have to change the script ID manually.

   If you do not change the script ID for a copied system script, the system script is removed from the list of scripts for this scenario and the new script is added as a scenario script. This lets you experiment with changes to a system script without actually changing it. The original system script will still be available to other scenarios.

4. Edit the script.
5. Click Duplicate. The script is added to the list.

36.9. Deleting a Script

If you delete a script that is assigned to a simulation object in a scenario, a task assignment, whether as an independent task or as part of a plan, will fail. If a scripted task is used as a subtask in another scripted task, that scripted task will fail. Scripted sets also fail. VR-Forces does not warn you if either of these cases exist. It is your responsibility to consider the effects of deleting a script.

➤ **To delete a script:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the script that you want to delete.
3. Choose **Script > Delete Script**.

36.10. Creating a System Script

To create a system script, you create a scenario script and then save it as a system script. System scripts are saved in *./data/simulationModelSets/<model_set>/scripts*. A system script has two files. The metadata is saved in a file ending with the extension *.xml*. The code is saved in a file with the extension *.lua*.

➤ **To save a scenario script as a system script:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. Select the script that you want to save as a system script.
3. Choose **Script > Promote to System Script**. You are prompted to confirm the change.
4. Click Yes.

36.10.1 Including System Scripts on the Task and Set Menus

Most of the system scripted tasks and sets included with VR-Forces are listed on the Task and Set menus. The scripts in the Examples folder and a few others are not listed. You can specify whether particular scripted tasks or sets are listed. You might want to do this to reduce clutter on the menus. Or you might want to limit the tasks available to players in a particular deployment of VR-Forces.

A system script's inclusion on a menu is controlled by these settings:

- The Show Task on Task Menu (or Show Set on Set Menu) check box on the New Script and Edit Script dialog boxes.
- The check box next to the script's entry on the Scripts dialog box (Figure 271).
- Its membership in a behavior set. The effect of behavior sets on script availability is additive to the configuration discussed in this section. For details about using behavior sets, see "29.6. Using Behavior Sets to Manage Scripts" on page 577.
- Crowd tasks file names must be included in the list of script IDs in *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/pedestrian-crowd-movement.sysdef*. The script files must be saved in *./data/simulationModelSets/EntityLevel/scripts*.

Figure 271. Show script on menu check box

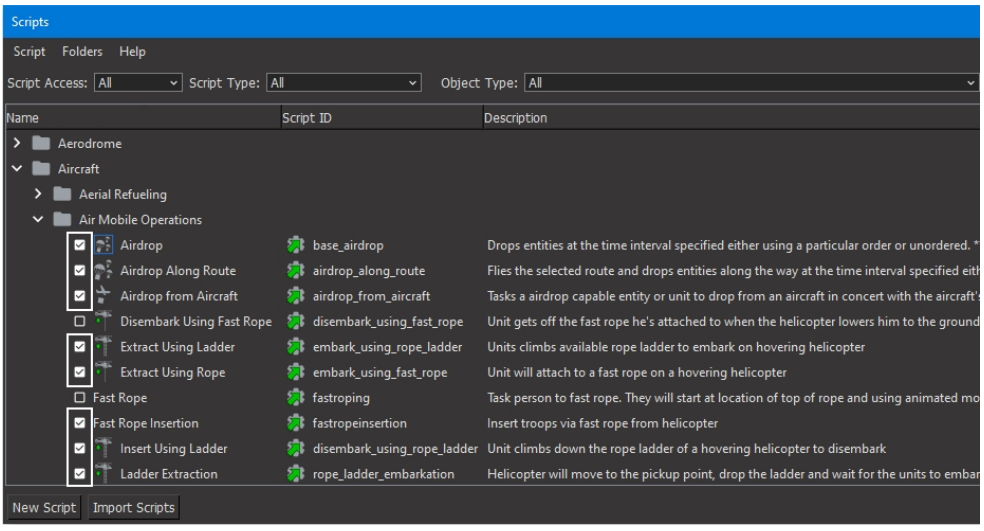


Table 37 shows when a scripted task is listed on a menu based on the configuration check boxes selected.

Table 37. System scripted task inclusion on Task menu

Show Script on Menu check box	System script list check box	Task included on menu
☑	☑	Yes
☑	☐	No, for this scenario
☐	☐	No

You cannot select the check box in the Scripts dialog box if the check box in the New Script or Edit Script dialog box is not selected. If you change the setting of the check box in the Edit Script dialog box, that change is saved in the script when you click Update.

If you unselect the check box in the Scripts dialog box, the task or set is immediately hidden from the menu. If you reload the scenario without saving first, however, it is restored to the menu.

➤ **To specify the availability of a system script on a menu:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256 on page 748).
2. In the list of system scripts, select the script whose availability you want to change. Use Table 37 to determine which check box configuration you need for your desired menu status.
3. Choose **Script > Edit Script Meta Data**. The System Script Editing prompt opens. (If you have disabled this prompt, the Edit Script dialog box opens. Skip to step 5.)
4. In the System Script Editing prompt, click Yes. The Edit Script dialog box opens.
5. Select or clear the Show Task on Task Menu check box (or the equivalent for sets).
6. Click Update.
7. In the list of scripts, select the check box next to the system script you are editing to make it available. Clear the check box to make it unavailable.
8. If this is for a crowd task, edit *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/pedestrian-crowd-movement.sysdef* and add the task to the **script-ids** list.

36.11. Specifying a Script Editor

ZeroBrane Studio is included with VR-Forces. VR-Forces generates a configuration file to support code completion and syntax highlighting features for all of the VR-Forces Lua API functions. However, you can use any text or script editor that you want.

➤ **To specify the editor to use for scripts:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Script Settings page (Figure 262 on page 764).

3. Click the Browse button () next to the Script Editor box. The Choose Script Editor dialog box opens.
4. Select the application that you want to use as the editor.
5. Click Open.

You can also open the Script Settings page from the Scripts menu in the Scripts dialog box.

36.12. Editing Lua Files

To edit the code for scenario scripts you need to work through the Script GUI because the scripts are embedded in the scenario scripts file rather than being saved as individual Lua files. System scripts are saved as Lua files, but even in this case it is better to edit them in the context of the entire script process rather than by editing the Lua file. However, VR-Forces includes some libraries of Lua functions that are independent of any of the scripts. They are available to be included in your scripts. If you want to edit the functions in these libraries or add to them, you need to edit the individual Lua files. The function libraries are in *./userData/scripts*. You can open them in any editor, or access them through the Scripts dialog box.

➤ **To edit a Lua file from the Scripts window:**

1. Choose **Simulation > Scripts**. The Scripts window opens (Figure 256).
2. Choose **Script > Edit External Script**. A file chooser dialog box opens.
3. Select the script that you want to edit. The file opens in a text editor.
4. Edit the file.
5. Save the file.



37. Writing Scripts

This chapter explains how to write Lua scripts for scripted tasks and sets.

37.1. The VR-Forces Lua Interface 784
37.1.1 Lua Classes 784
37.1.2 Lua Modules 786
37.2. Script Loading and Execution 787
37.2.1 Script Entry Points 787
37.2.2 The Script Execution Sequence 790
37.2.3 Limitations for Checkpointing Scripts 793
37.2.4 Editing Scripts While a Scenario is Running 795
37.3. Tasks and Subtasks 795
37.3.1 Monitoring the Status of Tasks and Subtasks 796
37.3.2 Stopping Tasks 797
37.3.3 Task Parameters 797
37.4. Geometry 801
37.4.1 Location3D 801
37.4.2 Vector3D 801
37.4.3 VectorOffset3D 803
37.4.4 VectorGeoc3D 803
37.5. Coding Messages for Translation 804
37.6. Error Detection 804
37.7. Interactive Scripts 805
37.8. A Basic Scripted Task 806
37.8.1 Create and Move to Waypoint Metadata 806
37.8.2 The Create and Move to Waypoint Lua Script 809
37.9. A Simple Scripted Set 810
37.10. A Simple Reactive Task 811
37.11. Background Processes 812

37.1. The VR-Forces Lua Interface

The VR-Forces Lua interface provides predefined elements that you can use to create scripts. These elements include object classes that represent the simulation, simulation objects, and geometric objects, plus Lua modules that provide tools for common tasks.

The classes and functions in the Lua interface are documented in the VR-Forces Lua API Documentation. This is a set of HTML files similar to the VR-Forces Toolkit documentation. It is accessible from the Documentation folder in the Windows Start menu or in *./doc/luadoc/index.html*.

37.1.1 Lua Classes

The Lua interface defines classes for common objects and functions. The classes in the interface are not fully featured classes as in the C++ language, but they do abstract the representation of information associated with an object and collect a set of methods related to the object. In Lua, to call a method `fn` on object `obj`, write:

```
obj:fn()
```

Several of the classes defined in the interface do not have constructors, because their objects are pre-defined or created only as return values to other functions. The classes and pre-defined objects in the interface are:

- *ComponentSystem*. A component system is a capability that is assigned to a simulation object in the Simulation Object Editor. Systems represent movement and damage capabilities, and represent individual weapons and sensors. In scripts, system objects are obtained from the ownship simulation object ("this") with a call to `getSystem()`, passing the name of the system (as used in the Simulation Object Editor) as an argument. The script can then access various attributes of the system through getAttribute calls.
- *FeatureSet*. This class represents a set of features obtained from a query to the terrain database. A *FeatureSet* is created with a *vrf* query function. *FeatureSet* methods allow a script to examine the features to determine their attribute values, vertex locations, and so on.
- *Location3D*, *Vector3D*, *VectorGeoc3D*, *VectorOffset3D*. These classes represent geometric objects. For more information, see "37.4. Geometry" on page 801.
- *ScriptAnimationModel*. *ScriptAnimationModel* refers to an animation model that can be used by a scripted task to animate an entity's movement. This class is serialized to the simulation and might contain movement data if supplied by the developer.
- *SensorGimbal*. *SensorGimbal* is a class of Lua object that represents a sensor that rotates on a gimbal.
- *SimObject*. *SimObject* refers to a single object in the simulation. It may be an entity, waypoint, route, or any other simulation object. These objects may be VR-Forces simulated objects, or remote objects received over DIS or HLA by VR-Forces. There is no constructor for SimObjects in the interface, but there are *vrf* functions that create objects, that get objects by name, that get nearby objects, and so on. *SimObject* methods are provided for getting information about the object such as

location, force type, and embarkation status.

- *SwingingArtPart*. *SwingingArtPart* is a class of Lua object that represents a rotating articulated part.
- *TranslatableArtPart*. *TranslatableArtPart* is a class of Lua object that represents a translating articulated part.
- *TranslatableStream*. *TranslatableStream* is a text stream class that can be translated by VR-Forces for display in the GUI as text that is translated to the current language the user is displaying (as long as those translations exist.)
- *this*. *this* is an object of class *SimObject* that is defined for all Lua scripts. It represents the simulation object running the script, also known as "ownship". It is not possible to construct a *this* object in a script. In addition to being able to use all of the *SimObject* methods, a Lua script can change the location and heading of *this*, manipulate its resources, and examine the contacts detected by the simulation object's sensors.
- *vrf*. *vrf* is defined as a global object for all Lua scripts. It is an instance of a class that represents the Lua interface; other instances cannot be created in a script. The *vrf* object provides functions that control and access information in the VR-Forces simulation.

In Lua, variables assigned to objects actually just hold a reference to an object, so that if one variable is assigned to another they will both hold a reference to the same object. In this example, `firstObj` and `secondObj` refer to the same object.

```
firstObj = getObjectByName("APC 1")
secondObj = firstObj
```

In most cases this is what a Lua programmer wants. However, for the geometric objects, a Lua programmer may sometimes want to make a copy of an object. A simple assignment does not make a copy in Lua, so the geometric objects have `makeCopy()` methods to provide copies.

Reserved Words

Table 38 lists reserved words (in addition to those reserved by Lua), which should not be used except for their intended purpose.

Table 38. Reserved words

Name	Module	Function
vrf	vrf	Module singleton
this	SimObject	Ownship name
ScriptVrfObject	SimObject	Class name
Location3D	Location3D	Class name
Vector3D	Vector3D	Class name
VectorGeoc3D	VectorGeoc3D	Class name

Table 38. Reserved words (continued)

Name	Module	Function
VectorOffset3D	VectorOffset3D	Class name
ScriptFeatureSet	FeatureSet	Class name
ScriptAnimationModel	ScriptAnimationModel	Global function
vrfprint	vrfutil	Global function
printDebug	vrfutil	Global function
printVerbose	vrfutil	Global function
printInfo	vrfutil	Global function
printWarn	vrfutil	Global function
printError	vrfutil	Global function
EntityType	vrfutil	Global table of functions
bhaveLoaded	vrfutil	Global function
new	fsm	Function name

37.1.2 Lua Modules

The Lua interface includes Lua modules for use in Lua scripts. You can access a module `stuff` in a Lua script by writing:

```
require "stuff"
```

The Lua documentation includes detailed information about each module. The modules available in the interface include:

- *aggregateUtils* is a collection of tables and functions that provide useful capabilities and definitions related to aggregates.
- *airportDataUtils* includes functions that compute information of interest about airports and runways.
- *behaviorEngine* implements a behavior engine aimed at supporting unit behaviors, or "commands." These commands are invoked and executed the same as all other Lua task scripts.
- *displayGraphUtil* has functions to assist with generating graph commands in the dot language and displaying the graph.
- *fileUtils* contains utility functions for reading data from text files.
- *formationUtils* includes functions used for coordinating tank platoons.
- *fsm* defines a class that implements finite state machine logic. There is one function in this module, `new()`, which creates an *fsm* object.
- *msgUtil* is a collection of functions that maintain queues of object or text messages that the entity receives.

- *munitionLauncherUtils* contains some functions that support munition-launching scripts such as Delayed_homing and Launch_ASW_Missile.
- *parseUtil* includes functions that assist with parsing of strings, especially from received messages.
- *pathUtil* is a collection of utility functions that help with path planning and ground-clamping points on a route.
- *spatialUtil* includes functions that compute things in space using points, vectors, and so on.
- *swgs_test* contains utility functions for SWGS performance test scenarios.
- *vrfutil* is not technically a Lua module. (It is not defined with a module function.) It is simply a set of global utility functions. The Lua script template that opens when a new script is created contains:

  ```
  require "vrfutil"
  ```

  so that these functions are available to the script by default.

The *.lua* module files are located in *./makLua*. You can add user-created modules to this directory and then import them into scripts with a `require` statement.

Lua also searches the *./userData/scripts* directory, therefore you can also save files there.

37.2. Script Loading and Execution

This section describes script entry point functions and the execution sequence for scripted tasks and sets. It also describes how VR-Forces checkpoints scripted tasks and executes them when a checkpointed scenario is run.

37.2.1 Script Entry Points

The Lua API includes functions that we call "entry points" to a scripted task. When you create a new script, they are included in the default script template. Unlike the Lua functions that you might write as part of a script, you do not have to call entry point functions from within the script. They are called by the simulation engine. (However, if an entry point function is not defined in a script, it is not called.)

- init(). Use init() to set up the initial state of your scripted task's environment. The init() function is the first function called when a scripted task is executed. It is called when a task is assigned to a simulation object, even if the scenario is paused. This function is called only once for an instance of a scripted task or background process when it starts. If a scenario is loaded with a scripted task or background process already started, the init() function is not called again because of the scenario load or rewind.
- tick(). The tick() function is the heart of every scripted task. It is called at every tick of the simulation engine once the simulation object has completed its initialization. Most of a scripted task's logic should be written into the tick() function as it is the

only entry point function consistently called in the script.

- suspend(). The suspend() function executes when a reactive task is triggered for a simulation object and the reactive task cannot run concurrently with the current task. The typical behavior, as embodied in the scripted task template, is to stop all tasks and subtasks.
- resume(). The resume() function executes when a reactive task completes. The typical behavior is to reinitialize the suspended task.

  > As you can see from their descriptions, the suspend() and resume() functions do not exactly suspend and resume tasks as those terms are commonly understood. In most cases (this is task dependent), VR-Forces does not preserve the state of a simulation object's tasks when they are suspended, it stops the tasks. Nor does it resume the previous task from the point at which it was suspended, it restarts the task from the current state of the simulation object. However, VR-Forces always saves the lua state of scripted tasks, so when a scripted task resumes, its variables may need to be reset in init() or checkInit().

- saveState(). If a script is running, the saveState() function is called just before a scenario is about to be saved. VR-Forces automatically saves the state of script execution. (For details, see "37.2.3 Limitations for Checkpointing Scripts" on page 793.) This function lets you do any additional processing to save the state of your scripted task. Most scripts do not use this function.
- loadState(). If a script was running when a scenario was saved, the loadState() function is called just after a scenario is loaded or rewound. VR-Forces automatically restores the state of a scripted task that was running when a scenario was saved. This function lets you do any additional processing required to restore the state of the script. If your script uses the saveState() function, you probably want to include complementary processing in loadState(). Most scripts do not use this function.

  > If your loadState() function assumes the presence of other simulation objects, you must take into account that other simulation objects might not have loaded yet when the function is called.

- shutdown(). The shutdown() function is called when the task is ending for any reason. You will not typically need to write any code in this function, however it is available if there is a case in which you want to do anything before the task ends.
- receiveObjectMessage(). The receiveObjectMessage() function is called when the simulation object executing this task receives an object message. An object message, in Lua, is a table with key-value pairs. The keys are strings, and the values may be numbers, strings, Booleans, tables, simulation objects, locations, vectors, or offset vectors. The function provides some data to determine what type of object message it is and which simulation object the message came from. Developers may write C++ code to send and receive these object messages, as well.

  This function takes three arguments:

  - **messageType**. A text string identifying the message that matches the registered name of the object message type. The types are pre-defined in VR-Forces.
  - **message**. A table of parameters that is specific to the messageType.
  - **sender**. The simObject that sent the message.

  Refer to the sendObjectMessage() function in the Lua documentation for messageType-specific parameters. A message of type "generic-table" may contain any keys and values; a script processing this type of message must examine the fields of the table to see what is in it.

- receiveTextMessage(). The receiveTextMessage() function is called when the simulation object executing this task receives a text message. The function provides some data to determine what the text message is and which simulation object the text message came from. This function takes two arguments, the message and the sender. The message is a text string and the sender is the simObject that sent the message.
- entityDestroyed(). The entityDestroyed() function is called once when the entity executing the script is destroyed.
- receiveInfluence(influencer, influenceName, influenceParams). The receiveInfluence () function is called in aggregate-level scenarios when a simulation object receives an Influence interaction from another simulation object. Influence interactions represent different ways an object might affect another. These include engagement, jamming, providing supplies, improving engineering objects, or destroying engineering objects.

  The receiveInfluence() function has the following parameters:

  - **influencer**. The simulation object that is performing this influence action.
  - **influenceName**. The name of the influence being performed. It matches one of the subclasses of the Influence interaction FOM class.
  - **influenceParameters**. A table of parameters of the influence. These parameters correspond to the parameters available to the Influence interaction FOM subclass, as specified by **influenceName**. Different subclasses have different parameters.

The following script entry points are only used in reactive tasks:

- check(). The check() function defines the condition under which the reactive task will trigger. It keeps checking the condition until it becomes true, at which point it starts the reactive task.
- checkInit(). The checkInit() function initializes the reactive task when it is enabled. It sets the initial conditions for the check() function. checkInit() runs when a reactive task is first enabled and runs again after the reactive task completes so that it can begin checking the condition again.

37.2.2 The Script Execution Sequence

A Lua script has several distinct sections, including the entry points described in "37.2.1 Script Entry Points" on page 787 and the global commands in the script. When a scripted task is assigned to a simulation object, the following actions happen, even when the scenario is paused:

1. The script environment is initialized. This step defines the *vrf* and *this* objects and all of the class functions in the Lua API.
2. The script text is loaded. All Lua statements in the global scope are executed. Normally, the script defines (assigns initial values to) global variables here.
3. The `init()` function is called.

This sequence is followed whenever a scripted task is assigned, including these situations:

- A scripted task is assigned while the scenario is running.
- A simulation object has a script assigned at the beginning of a scenario and the scenario is loaded or rewound.
- You modify the script or script metadata while the script is running.

If the simulation is running, after the `init()` function is called VR-Forces begins calling the `tick()` function at the period specified by the `setTickPeriod()` function (default: 0.5 seconds).

Suppose a script "test" contains the code:

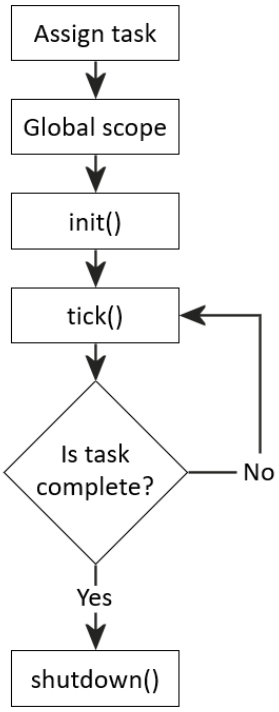
```
print("Global statements")
t = {}
function init()
  print("Init function")
  vrf:setTickPeriod(0.5)
  t[1] = 0
end
function tick()
  table.insert(t, vrf:getExerciseTime())
  print("Tick ", #t)
end
```

If the script is assigned to a simulation object and then the scenario runs for a short time, the console output would be:

```
Global statements
Init function
DtCgfDispatcher::controlScenario: run
Tick 2
Tick 3
Tick 4
DtCgfDispatcher::controlScenario: pause
```

Figure 272 illustrates the control flow for a scripted task.

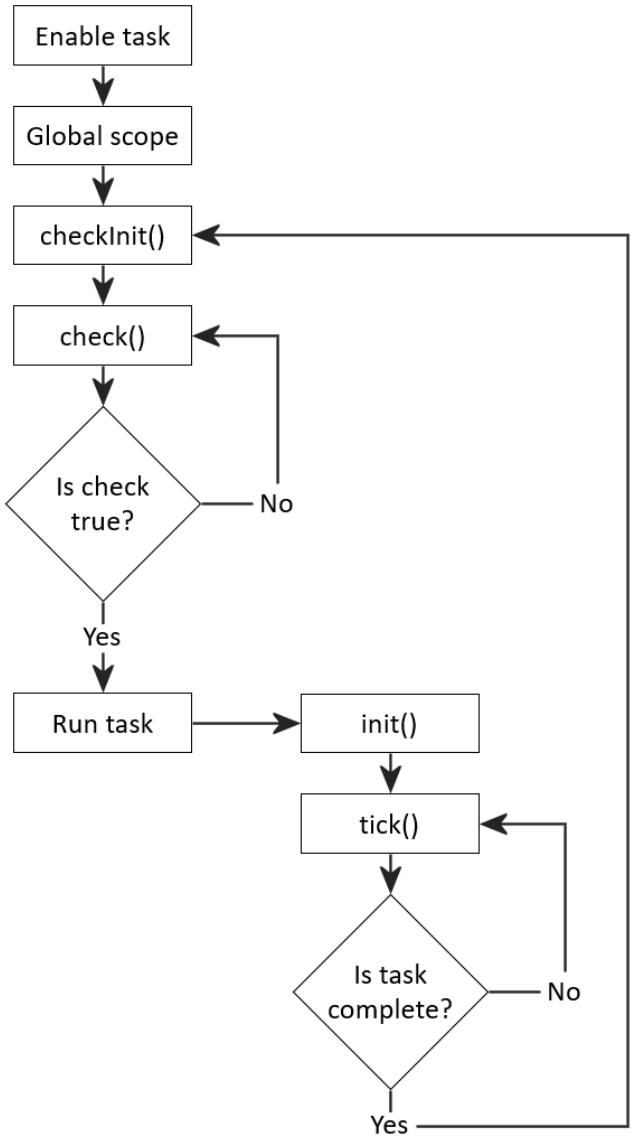
Figure 272. Scripted task control flow



The Script Execution Sequence for Reactive Tasks

Figure 273 illustrates the script execution sequence for reactive tasks. It is distinguished from that for scripted tasks by the addition of the checking process.

Figure 273. Reactive task control flow



The Script Execution Sequence for Saved Scenarios

When a scenario is saved (checkpointed), VR-Forces first runs `saveState()` and then saves the Lua state in the scenario file. (Checkpointing a scripted task has some limitations. For details, see "37.2.3 Limitations for Checkpointing Scripts" on the facing page. For general details about checkpointing, see "7.3. Saving a Scenario" on page 247.) When a saved scenario is loaded, the following steps are taken to restore the scripted task:

1. The script environment is initialized. This step defines the *vrf* and *this* objects and all of the class functions in the Lua API.
2. The script text is loaded. All Lua statements in the global scope are executed. Global variables defined and initialized in this part of the script are re-initialized.
3. VR-Forces restores the values of global variables to the saved values. These values replace the initial values assigned in step 2.
4. The `loadState()` function is called.

> The `init()` function is not run when a scenario is reloaded.

Assume that the "test" scripted task described in the previous section is saved at the point shown in the sample code (`t = 4`). When the scenario is run from the checkpoint, the console output will be:

```
DtCgfDispatcher::controlScenario: run
Tick 5
Tick 6
DtCgfDispatcher::controlScenario: pause
```

37.2.3 Limitations for Checkpointing Scripts

When VR-Forces saves (or checkpoints) a scenario, it saves the following aspects of the Lua state in the checkpoint:

- The values of global variables that contain strings, numbers, and objects (of the classes listed in "37.1.1 Lua Classes" on page 784).
- Tables, but only table values that contain strings, numbers, objects, and tables. Tables cannot contain values that refer to the same table, either directly or indirectly through another table.

The following aspects of the Lua state are not saved in a checkpoint:

- Variables containing functions or threads.
- Variables declared in the global scope with the keyword **local** preceding the declaration.
- Table entries that do not use numbers or strings as keys.

Lua scripts that contain elements that are not saved can still be restored after a checkpoint if the global statements in the script reset the elements. For example, a variable containing a function is not saved in the scenario file, but if this variable is initialized in the global Lua statements, it is reset when those statements are loaded. However, if the value of this variable was changed during script execution (in the init() or tick() functions), then its value will not be restored correctly.

Tables that reference themselves in the table will not save properly.

To demonstrate how scripted tasks are saved, assume the following script:

```
print("Global statements")
```

```
a = 0
t = {foo = 0, bar = 1}
f = function () print("hello") end
function init()
  print("Init function")
  vrf:setTickPeriod(0.5)
  a = 100  -- change a
  t.foo= 10 -- change t.foo
  t.bar = nil -- remove t.bar
  f = function () print("world") end -- change f
end
function tick()
  print ("Tick. a = ",a, "t.foo = ", t.foo, "t.bar = ", t.bar)
  f()
  print()
end
```

The output the first time the scripted task runs is:

```
Global statements
Init function
DtCgfDispatcher::controlScenario: run
Tick. a = 100     t.foo = 10      t.bar = nil
world
```

When the scenario is saved and then reloaded, and the simulation is run again, the following is printed:

```
DtCgfDispatcher::controlScenario: run
Tick. a = 100     t.foo = 10      t.bar = 1
hello
```

A comparison of the outputs reveals that variable **a** and table entry **t.foo** were restored to the correct values. However, table entry **t.bar** and function **f** were re-initialized in the global statements, but were not restored to the values they had when the scenario was saved.

Saving the State of Global Variables

Global variables are saved when a scenario is checkpointed in one of the following ways:

- If the checkpoint mode is AllGlobals, all global variables defined are saved.
- If the checkpoint mode is CheckpointStateOnly, the script only saves variables that are part of the checkpointState table. If you remove this value, it defaults to the behavior of saving all globals.

If you want to change the mode, call setCheckpointMode(AllGlobals) to save all globals or setCheckpointMode(CheckpointStateOnly) to save only those variables in the checkpointState table. They are reinitialized when a checkpointed scenario is loaded.

```
vrf:setCheckpointMode(CheckpointStateOnly)
```

The following code, from *vrfutil.lua*, defines some global values:

```
-- Global booleans to help define which checkpoint state to use
AllGlobals = false
CheckpointStateOnly = true
```

For more information about vrf:setCheckpointMode, see the VR-Forces Lua documentation.

37.2.4 Editing Scripts While a Scenario is Running

You can add, remove, and edit scripts during scenario creation or while a scenario is running. If a script is not being used by any simulation objects, editing or deleting it has no effect on the scenario. If you change a script that is being executed by a simulation object, when you save the change the script may be stopped, restarted, or continue to execute depending on what has changed.

Table 39 lists the effect on scripts and scripted tasks running as subtasks if you edit the code or metadata:

Table 39. Side effects of editing scripts

Change to script	Result
Change code or any metadata except the script ID. Entity type is still supported.	The script restarts. Changes to the Action Category take effect when the scenario restarts.
Entity types supported changes.	The script stops for simulation objects that no longer support it.
Script ID changes.	The script continues to execute. Any script that uses this scripted task as a subtask will fail until its code is updated to use the new script ID.

If you changed the simulation object types that can use the script, then it is added to or removed from the menu for the appropriate simulation object types. (This change takes place immediately.)

37.3. Tasks and Subtasks

The primary way that a scripted task carries out actions is through starting subtasks for the simulation object running it or sending tasks to other simulation objects. All C++ tasks and other scripted tasks are available for sending as subtasks or tasks.

If a scripted task needs to give a task to the simulation object that is running it, it does so as a subtask. (The simulation object's current task is the scripted task itself, so any action that it needs to take is a subtask to the scripted task.) Subtasks are started using the `vrf:startSubtask()` function.

If a scripted task needs to give a task to a simulation object other than the one running it, it uses the `vrf:sendTask()` function.

A `startSubtask()` or `sendTask()` function specifies the task name and task parameters, if any. The name must be the name of one of the C++ tasks, as listed in the Task page of the Lua API Documentation, or the script ID of another scripted task.

The task parameters are passed as a Lua table, with parameter names as the indices and the parameter values as the values. The valid parameters for C++ tasks are in the Lua API documentation. For scripted tasks, the parameter names are the parameter names defined for that task. You can view them by opening a scripted task in the Edit Script dialog box. Any parameters that are not specified when starting the task use their default values.

This example subtask gives a Move To task to the simulation object running the scripted task, specifying a waypoint:

```
vrf:startSubtask("move-to", {destination = "Waypoint 1"})
```

If a task does not have any parameters, `startSubtask()` must represent the parameters as `{}`. For example:

```
vrf:startSubtask("wait", {})
```

This example task sends a Patrol Between Waypoints task to the simulation object named "someEntity":

```
vrf:sendTask(someEntity, "patrol-two-points", {first_control_
point="Waypoint 1", second_control_point="Waypoint 2"})
```

If a task starts with an invalid name or invalid parameters, a runtime error occurs in the script.

Whenever a task or subtask starts, a task ID integer is returned. This task ID is used to monitor or stop the task.

If a task is started on a simulation object that cannot execute it, the task will enter the Complete (failed) state shortly after the start call is made. This may not happen until a subsequent tick of the scripted task.

37.3.1 Monitoring the Status of Tasks and Subtasks

Once a task is started, the scripted task that started it can monitor it using its task ID. A task can be in one of three states:

- Running. The task was started, but has not completed or been canceled.
- Complete (success). The task has completed, and called endTask(true) to end itself.
- Complete (failure):

  - The task has ended itself by calling endTask(false).
  - The task failed to start because the simulation object was incapable of executing this task.
- Canceled:
  - The task was explicitly canceled using stopTask().
  - The task was replaced by a conflicting task.
  - The user issued a Skip Task.

> The success or failure of task completion is defined by the task being executed, and may not be consistent between task types.

Subtasks are monitored with the functions:

- **isSubtaskComplete(taskID)**
- **subtaskResult(taskID)**
- **isSubtaskRunning(taskID)**
- **isSubtaskCanceled(taskID)**.

Tasks are monitored with the functions:

- **isTaskComplete(taskID)**
- **taskResult(taskID)**
- **isTaskRunning(taskID)**
- **isTaskCanceled(taskID)**.

37.3.2 Stopping Tasks

Any task or subtask that has been started from a scripted task can be stopped by that task before it has completed. This is done using **stopTask(taskID)** or **stopSubtask(taskID)**.

37.3.3 Task Parameters

Task parameters are the parameters that are gathered from the user, or set by a parent task, when a task is started. They are passed into the new task when it starts. A script gets the value of a parameter with the syntax **taskParameters.***parameterName*, as in this example:

```
  waypoint = vrf:createWaypoint
({location=taskParameters.waypointLocation})
```

For scripted tasks, the task parameters are available in a Lua table named taskParameters available in the global scope. The table contains items named using the parameter names specified in the parameters section of the Scripted Task dialog box when the task was created. The types of these parameters depend on the parameter types selected by the user in this dialog box.

Table 40 lists the parameter types you can specify when you create a scripted task and the corresponding Lua types in the taskParameters table.

Table 40. Task parameters and Lua types

Parameter Type	Lua Type
Active Sonar Mode	
Aggregate Formation	String.
Altitude Label	View only. No Lua type.
Altitude MSL	Number (meters).
Altitude Rate Change	Double (meters/second).
Animated Movement	String.
Bomb Resource	String (resource name).
Boolean	Boolean. Must be true or false (lowercase).
Choice (Drop-Down List)	Number (Index of the option chosen).
Choice (Image Drop-Down List)	Number (Index of the option chosen).
Choice (Multiple Option Buttons)	Number (Index of the option chosen).
Choice (Single Option Button)	Number (Index of the option chosen).
Color	Color: rgba (name red, green, blue, alpha from 0 - 255).
Date/Time	Integer (seconds in UTC).
Depth MSL	Double (meters).
DI-Guy Animation	String.
DI-Guy Appearance	String.
DI-Guy Gesture	String (gesture name).
Direct Fire Weapon	String (weapon name).

Table 40. Task parameters and Lua types (continued)

Parameter Type	Lua Type
Distance	Number (meters).
Embarkation Slot	Slot: <name>.slot, <name>.chooseAny (where slot is an integer and chooseAny is a boolean which means choose any other if that slot is not available.)
Embedded Entity	String (embedded entity name).
Emitter	Number (Index of the emitter system).
Entity Type	String (entity type as x:x:x:x:x:x:x).
Entity Types	Table (list) of entity types.
Filename	String (pathname).
Simulation Object Type	String (DIS type enumeration, colon separated).
Force	String.
Heading	Number (radians).
Heading Label	View only. No Lua type.
Indirect Fire Weapon	String (name of weapon).
Integer	Number.
Integer Map	Table of key of string and value of integer.
Jammer Selector	Interger (jammer selected, like Radar Mode).
Jamming Mode	Integer (mode selected, like Radar Mode).
Loadout	String.
Location (with altitude)	Location3D.
Location (without altitude)	Location3D.
Munition Resource	String (resource name).
Naval Mine Resource	String (resource name).
New Tactical Graphic	A table pf Dt:pcatopm3D Lua objects.
Observer Mode	String.

Table 40. Task parameters and Lua types (continued)

Parameter Type	Lua Type
Observer Saved View	A string that contains the XML ouptut of an observer view record.
Offset Location	VectorOffset3D.
Overlay Parent	String.
Radar Mode	Integer.
Radar Selector	Integer (radar selected, like Radar Mode).
Range Label	View only. No Lua type.
Rate Label	View only. No Lua type.
Real	Number.
Real Map	Map of key/value of key of string and value of double.
Resource	String (resource name).
Separator	Not applicable.
Simulation Object (Single)	SimObject.
Simulation Objects (Multiple)	Table of SimObjects.
Speed	Number (meters / second).
Sphere	Sphere: variable.center (Location3D), variable.radius (radius in meters).
String	String.
String Map	Map of key/value of key of string and value of string.
Time	Number (seconds).
Topographic Orientation	A Vector3D with bearing and inclination. Does not include roll.
Turn Rate	Number (radians / second).
Turn Rate Label	View only. No Lua type.
Visual Meta Data	String Map that contains key/value pairs of the marking and the value for the marking.
Weapon	String.

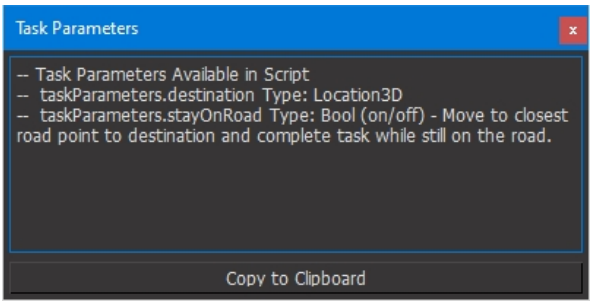
Getting the Task Parameters for a Script

You can quickly get a list of the parameters available in a scripted task or set, their type, and a description, if one was created. You can copy the list to the clipboard so that you can paste it into your script.

➤ **To get a list of the task parameters in a script:**

1. Choose **Simulation > Scripts**. The Scripts dialog box opens.
2. Select the script whose parameters you want.
3. Choose **Script > Edit Script Meta Data**. If this is a system script, you might be prompted to continue. If so, click Yes. The Edit Script dialog box opens.
4. Click the Show Task Parameters Code button (). A window opens that lists the parameters in the script.

Figure 274. Task parameters list



5. Optionally, click Copy to Clipboard.

37.4. Geometry

The Lua interface includes classes representing geometric objects. The classes represent locations and vectors in three dimensional space. Different classes are defined for vectors in different contexts so that functions that manipulate them know what their context is and can convert between contexts automatically. Therefore, a script does not have to, for example, create rotation matrices to perform coordinate transformations between vectors that are in an entity-body context and vectors that are in a world context.

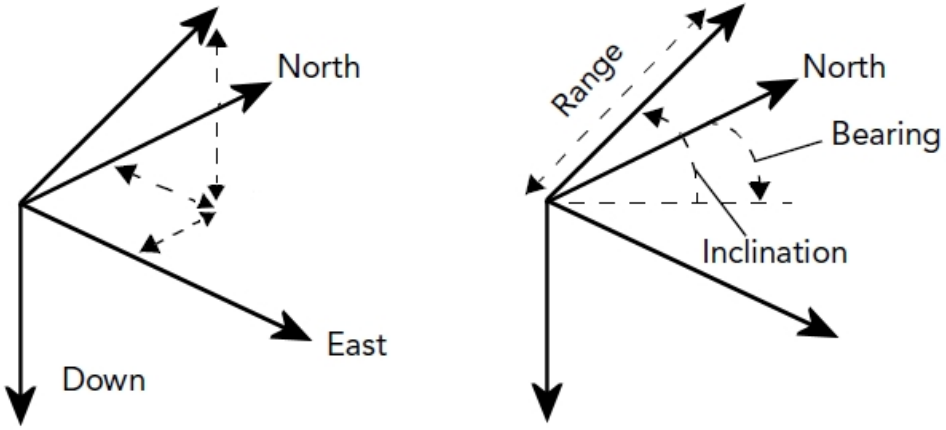
37.4.1 *Location3D*

A *Location3D* represents a point in three-dimensional (3D) space. Scripts can access the three components of the location as latitude, longitude, and altitude. There is no other access to components in any other coordinate system. Methods are available to construct a *Location3D*, to create a vector from the difference between two locations, and to generate a new point by adding a vector to a location.

37.4.2 *Vector3D*

A *Vector3D* is a direction in a north-east-down coordinate system. Scripts can access these three components of a *Vector3D*, or access bearing-inclination-range (Figure 275). This is the type of vector that scripts will use most often to find directions between locations, to create new locations relative to other locations, and so on.

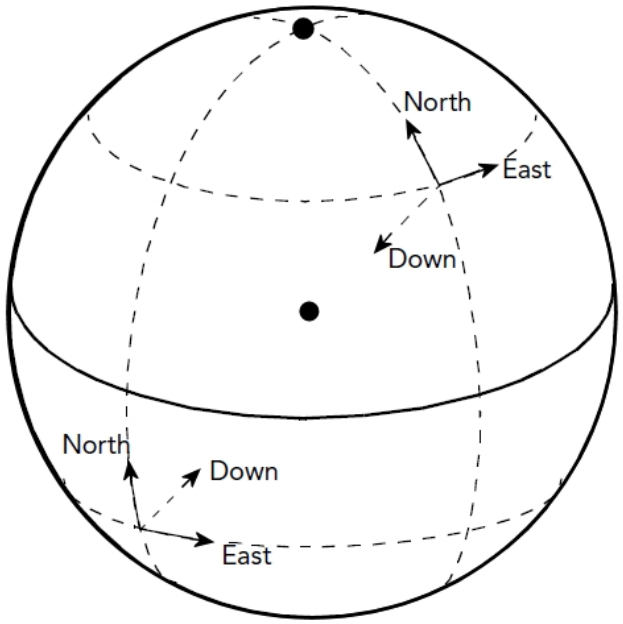
Figure 275. Vector3D: North-East-Down, or Bearing-Inclination-Range



When viewed from the perspective of a 3D Cartesian coordinate system that is fixed to the earth, there is no single north-east-down coordinate system. This is because "north" and "east" directions vary depending on where the observer is located on the earth. The *Vector3D* is therefore similar to a vector in topographic coordinates. At higher latitudes, "north" for two different simulation objects may not be the same direction (for example, two planes near the north pole, but at different longitudes, that are both flying "north," are not on parallel courses).

Figure 276 shows two north-east-down coordinate systems with origins at different locations on the earth. The coordinate systems have different orientations with respect to a coordinate system fixed to the earth.

Figure 276. Coordinate systems with different origins



A *Vector3D* is not strictly a vector, in that it is not a straight line in a Cartesian coordinate system fixed to the earth. Just as the direction "east" changes (viewed in the Cartesian coordinate system) to follow the curvature of the earth, so does a *Vector3D*. This definition is different from topographic coordinates, which are Cartesian coordinates fixed to a point on the surface of the earth. Implications of this definition include:

- The heading of a *Vector3D* from one location to another is the heading of the shortest route around the earth (the great circle route) between the locations. The length is the great circle distance.
- The inclination of a *Vector3D* between two locations with the same altitude is 0.0, even when one location is over the horizon from the other.
- A *Vector3D* with length equal to the circumference of the earth, and inclination 0.0, when added to a location on the surface of the earth will produce a location coincident with the first location.

37.4.3 *VectorOffset3D*

A *VectorOffset3D* is a set of three values that define an offset from some reference direction defined by a *Vector3D*. The three values represent *right-forward-up*, where:

- *forward* is in the same direction as the reference.
- *right* is to the right in a horizontal plane. Horizontal means the plane defined by an inclination of zero; this definition implies that the "roll" component of the reference direction is assumed to be zero.
- *up* is perpendicular to *forward* and *right*, that is, *right* X *forward*.

A *VectorOffset3D* is appropriate to use, for example, to define the offsets of simulation object formation positions from a central location. Actual formation positions could be generated by transforming the offsets to *Vector3Ds* using the direction vector of the simulation object, and then adding the *Vector3Ds* to the simulation object location.

37.4.4 *VectorGeoc3D*

A *VectorGeoc3D* is a vector in a Cartesian coordinate system fixed to the earth (geocentric coordinates). The actual coordinate meanings and values are not accessible to the script, because they are considered to be abstract. A *VectorGeoc3D* can be generated only by taking the difference of two locations, or by transforming a *Vector3D* at a particular location.

A *VectorGeoc3D* between two locations is the line-of-sight vector between the locations. Therefore:

- A *VectorGeoc3D* between two points on opposite sides of the earth, at the same altitude, will go through the earth. (A *Vector3D*, by contrast, will go around the earth parallel to the surface.)
- The angle between two *VectorGeoc3D* vectors corresponding to north for two simulation objects near the north pole will be the difference in their longitudes. (The angle between their *Vector3Ds* for north, by contrast, will be zero.)

The Lua interface does not have any simulation functions that require *VectorGeoc3Ds*. However, they may be manipulated to compute angles and so forth.

37.5. Coding Messages for Translation

VR-Forces has a utility, translationFileCreate, that can extract the user-visible text in a script dialog box and make it available for translation. (For details, see "2.6. Localizing the Graphical User Interface" on page 108.) If a script contains messages such as printWarn, printInfo, and so on, you can code these messages so that they are also extracted to the translation file.

To code messages for translation, use the syntax:

```
printWarn(vrf:trUtf8("Text message."))
```

You can include variable data in the message:

```
printWarn(vrf:trUtf8("Some text %1. Some more text %2):arg
(variable1):arg(variable2))
```

where an argument can be a string or number.

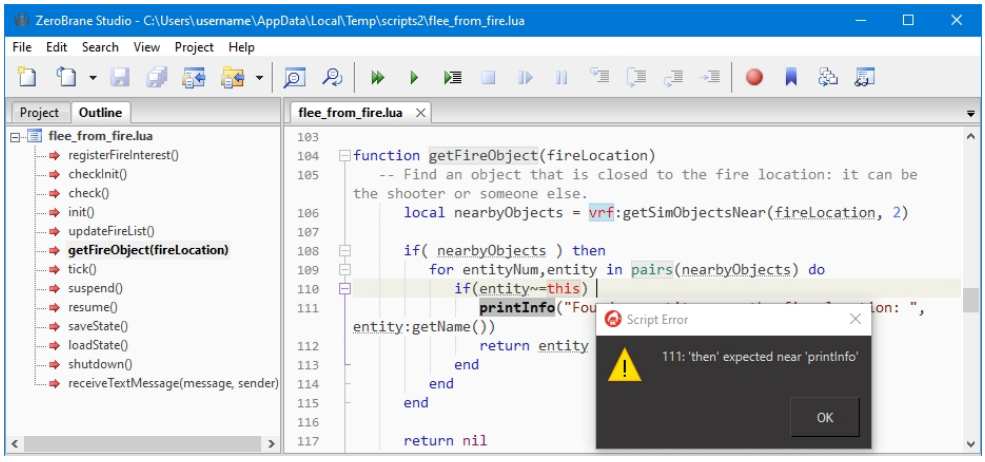
For more information about coding for translation, see the !HowToPrint page in the Lua documentation (*./doc/luadoc/index.html*).

37.6. Error Detection

The VR-Forces Lua implementation can detect syntax errors and runtime errors.

When you save a script, it is checked for syntax errors. If an error is found, it is reported in an error message. In Figure 277, the **if** statement in line 38 is missing the **then** keyword.

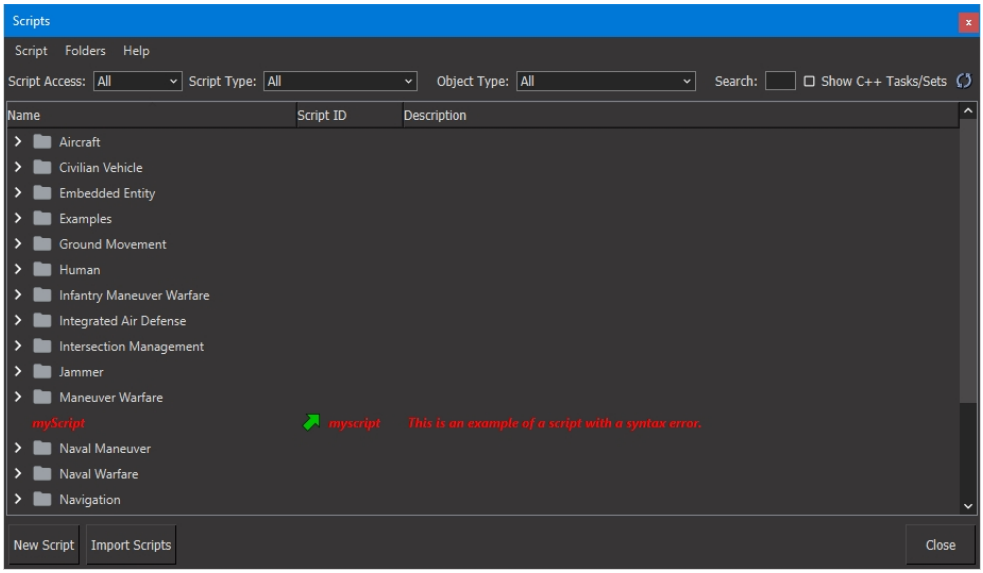
Figure 277. Lua syntax error



Detection of a syntax error does not prevent the script from being saved. If you do not fix the error and try to use the script, the sim engine generates an error.

When you run a script, the sim engine can detect errors due to incorrect use of user functions and the Lua API functions, such as calling an object that does not exist, or trying to access feature data before it is paged in. When a runtime error is detected, the sim engine generates an error message. When the sim engine generates an error, the script that caused the error is displayed in bold, red font in the Scripts dialog box (Figure 278). Therefore, you may want to keep the dialog box open while you are testing new or edited scripts.

Figure 278. Lua error message



➤ **To view an error message:**

1. In the Scripts dialog box, select the script that has generated an error.
2. Choose **Script > View Script Error**. An error message displays.

Script errors are also sent to the console in the Information dialog box.

37.7. Interactive Scripts

You can write scripts that request input from the VR-Forces user. Script questions are sent to the simulation object's object console and to the Object Console Summary Panel. Users can answer questions from the Object Console Summary Panel.

To create interactivity in a script, use the vrf:askUserQuestion() Lua function. When a script sends a question to the console, the Answer Questions button becomes active. When the user clicks the button, a set of choices is displayed. The choices can be presented as buttons, as option buttons, or as a drop-down list.

The script can do whatever it wants with the user input.

37.8. A Basic Scripted Task

VR-Forces includes many scripted tasks, all of which can serve as examples of how to write one. The Create and Move to Waypoint scripted task is a basic example that shows you how to:

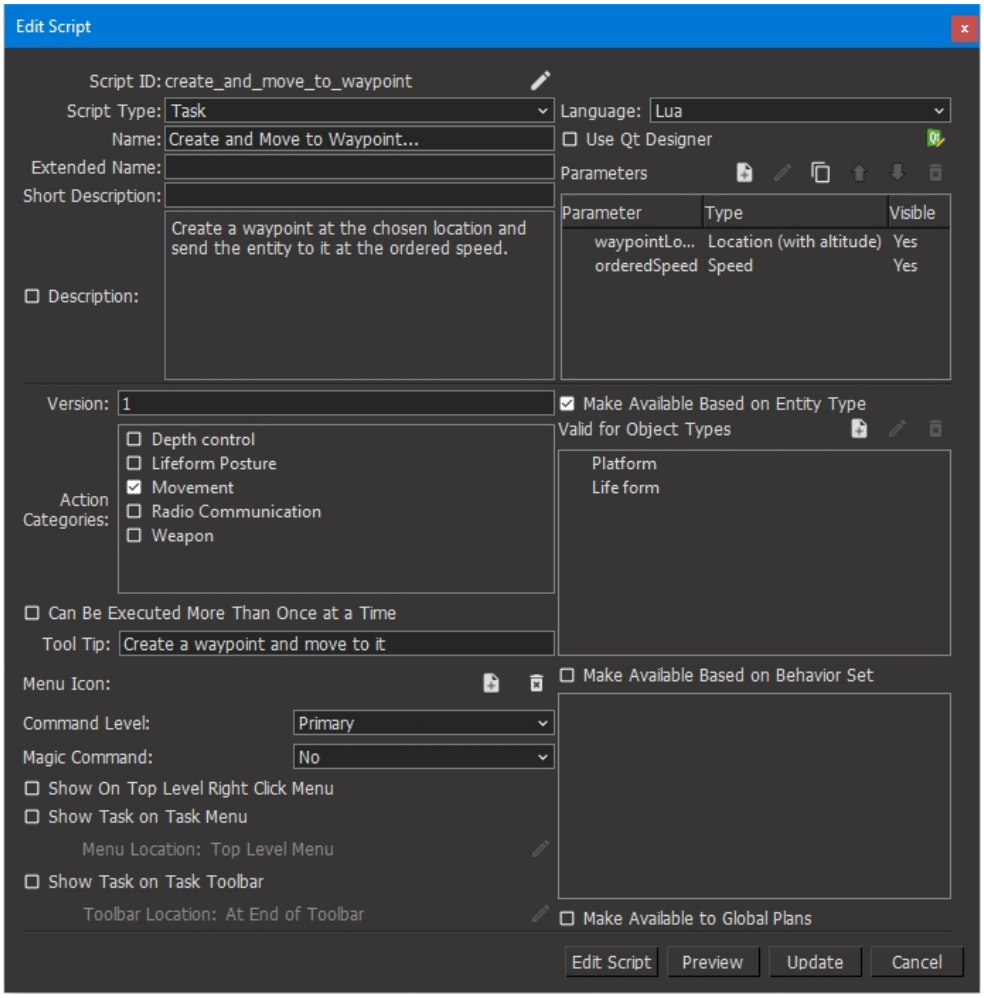
- Set up metadata.
- Include the VRF Utilities module.
- Initialize a subtask ID.
- Use the init() function.
- Create a waypoint using input from the task's dialog box.
- Send a set data request to a simulation object, using input from the task's dialog box.
- Use the tick() function.
- Send a subtask and get the subtask ID.
- Check for completion of the task.
- Stop the task.

The Lua code is comprehensively commented. This section supplements the comments.

37.8.1 Create and Move to Waypoint Metadata

Figure 279 shows the metadata for the Create and Move to Waypoint scripted task.

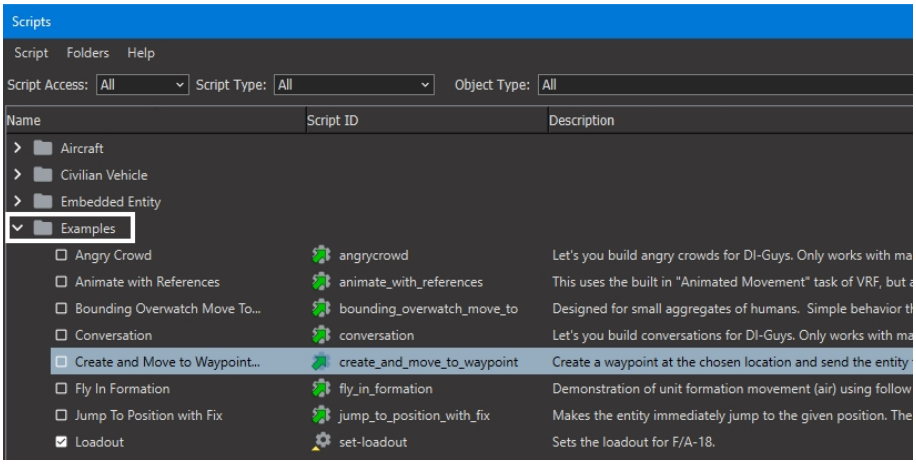
Figure 279. Create and Move to Waypoint script



➤ **To view the metadata:**

1. Choose **Simulation > Scripts**. The Scripts dialog box opens.
2. Expand the Examples folder (Figure 280).

Figure 280. Scripts dialog box



3. Select Create and Move to Waypoint.
4. Choose **Script > Edit Script Meta Data** or double-click the script name. The Edit Script dialog box opens (Figure 279).

Note how the Name column of the Create and Move to Waypoint entry in the Scripts dialog box matches the Menu Text in the Edit Script dialog box. The Script ID and Description fields are also drawn from the scripted tasks metadata.

Now let's look at the parameters. This scripted task has two parameters, waypointLocation and orderedSpeed. To see the definition of a parameter, select it and click the Edit button (). Figure 281 shows the dialog boxes that specify the parameters.

Figure 281. Parameter definitions

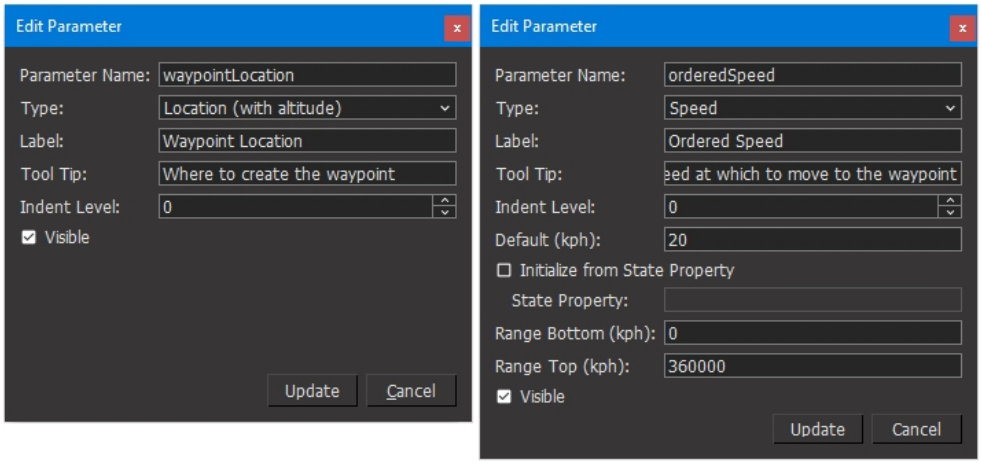
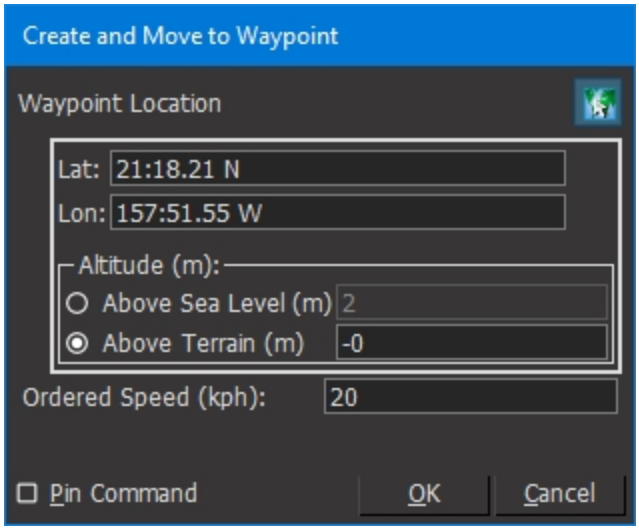


Figure 282 shows the dialog box that is displayed when you assign a simulation object this task. Note that the description of the task on the dialog box matches the description in the metadata. The parameter labels in the dialog box match the labels specified for the two parameters. The labels are not the same thing as the parameter name. The parameter name is the name used in the Lua script for this scripted task.

Figure 282. Create and Move to Waypoint dialog box



When this scripted task is assigned to a simulation object, the user will specify the waypoint and the ordered speed. Then the scripted task executes. Now we will look at the Lua script to see what happens.

37.8.2 The Create and Move to Waypoint Lua Script

The entire Lua script for this scripted task (without comments) is:

```
1  require "vrfutil"
2  moveToWaypointSubtaskId = -1
3  function init()
4      vrf:setTickPeriod(0.5)
5
6      waypoint = vrf:createWaypoint
   ({location=taskParameters.waypointLocation,
7          force=this:getForceType()})
8
9      vrf:sendSetData(this, "set-speed",
10         {speed=taskParameters.orderedSpeed})
11 end
12 function tick()
13     if moveToWaypointSubtaskId == -1 then
14         moveToWaypointSubtaskId = vrf:startSubtask("move-to",
15             {control_point=waypoint})
16     end
17     if vrf:isSubtaskComplete(moveToWaypointSubtaskId) then
18         vrf:endTask(true)
19     end
20 end
21 function shutdown()
22     vrf:deleteObject(waypoint)
23 end
```

The comments in the script describe the control flow. The essence is that, using the input from the task dialog box, the waypoint is created and the speed is set in init() (line 3). The first time the task is ticked, it sends the Move to Waypoint subtask. Thereafter, each time the task is ticked (line 12), it checks to see if the subtask is complete (line 13). When the subtask is complete (line 17), the task is ended and the waypoint is removed.

The functions in this script are all in the *vrf* class, which is described in "37.1.1 Lua Classes" on page 784. The createWaypoint() function (line 6) takes its location from the waypointLocation parameter that was defined in the scripted task metadata and whose value was provided in the Create and Move to Waypoint dialog box when the task was assigned. Similarly, the ordered speed is set with the sendSetData() function (line 9) and the value is taken from the orderedSpeed parameter for the task. The script gets the parameter values with the **taskParameters.***parameter* syntax.

This script uses three of the entry point functions – init(), tick(), and shutdown(). If you were to checkpoint the scenario while the script was executing, the standard save actions would take place, but no script-specific actions would be taken, because there is no saveState() function in the script. Similarly, no special action would be taken when the scenario was loaded again because loadState() is not used.

For details about the parameters and return values of the functions, please see the VR-Forces Lua API documentation. Similarly, the tasks and set data requests that you can assign with the startSubtask() and sendSetData() functions are described on the Task and Set pages of the Lua API documentation. You should review the lists of functions to see what you can do in a scripted task.

37.9. A Simple Scripted Set

This example script shows how to create a scripted set. It takes two input values from a dialog box using setParameters. Then it uses the sendSetData() function to set the values for the entity. It needs an endSet() function to finish the process.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
  vrf:setTickPeriod(0.5)

end
function tick()
  vrf:sendSetData(this, "set-speed", {speed=setParameters.setSpeed})
  vrf:sendSetData(this, "set-force", {force=setParameters.setForce})
  vrf:endSet()
end
```

The following scripted task does exactly the same thing as the scripted set. It sets the speed and force of the selected simulation object. The difference is that since this is a scripted task, it interrupts the task that the simulation object is executing. Try these two scripts to see the difference.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
```

```
    vrf:setTickPeriod(0.5)

  end
  function tick()
    vrf:sendSetData(this, "set-speed",
{speed=taskParameters.setSpeed})
    vrf:sendSetData(this, "set-force",
{force=taskParameters.setForce})
    vrf:endTask()
  end
```

The following scripted task does exactly the same thing as the scripted set. It sets the speed and force of the selected simulation object. The difference is that since this is a scripted task, it interrupts the task that the simulation object is executing. Try these two scripts to see the difference.

```
  require "vrfutil"
  function init()
    vrf:setTickWhilePaused(true)
    vrf:setTickPeriod(0.5)

  end
  function tick()
    vrf:sendSetData(this, "set-speed",
{speed=taskParameters.setSpeed})
    vrf:sendSetData(this, "set-force",
{force=taskParameters.setForce})
    vrf:endTask()
  end
```



As you can see, there is nothing in the scripts that identifies them as a scripted task or scripted set other than the endTask() and endSet() functions. VR-Forces determines the type of script from the myScriptType value in the metadata. Scripted task types are defined in *./include/vrfutil/scriptedTaskMetaData.h*.

37.10. A Simple Reactive Task

The difference between a scripted task and a reactive task is the use of the check() and checkInit() functions in the reactive task. The check() function tests a condition at whatever tick rate you specify. If the condition returns true, the task runs.

The simple reactive task in this section monitors the speed of a simulation object. When it exceeds 15 meters per second, the simulation object is destroyed. The script works as follows:

1. The checkInit() function sets the tick rate to 1.0 second.
2. In check(), VR-Forces gets the speed of the simulation object and prints it to the console.
3. It tests to see if the speed is greater than 15 meters per second. If the condition is met, the function returns true.
4. If the function returns true, the script executes init() and begins to tick().
5. In the tick() function, the simulation object is destroyed and the task is complete. Note that in the executeSetData() function, set-destroy does not have parameters, but the command must include empty braces.

```
function checkInit()
  vrf:setTickPeriod(1.0)
end

function check()
  currentSpeed = this:getSpeed()
  print("speed is:", currentSpeed)
  if currentSpeed > 15 then
      return true
  end

  return false
end

function init()
  vrf:setTickPeriod(0.5)
end

function tick()
  vrf:executeSetData("set-destroy", {})
  vrf:endTask(true)
end
```

37.11. Background Processes

Background processes are scripts that run continuously in the background. When you create a background process, you specify which simulation object types it is valid for. Thereafter, whenever you create those simulation object types the background process initializes as soon as the simulation object is created. The supported simulation object type can be as general or as specific as you want it to be, based on the entity type enumeration that you specify.

Background processes are used for simulation object actions that happen continuously in a scenario, such as consuming or receiving supplies, calculating the simulation object's footprint, and automatically attacking opposing forces.

Background processes have one important restriction - they cannot issue tasks to simulation objects or stop their tasks.

There is no indication in the GUI that a background process is running. The only way to know which background processes are available and which simulation objects they might be affecting is to examine them in the Scripts dialog box.

The following code is a very simple background process that checks a simulation object's speed every five seconds. If the speed exceeds three meters/second, it prints a message to the console. The script's metadata specifies that it applies to lifeforms. Therefore, this background process will run for each lifeform entity in the scenario.

```
require "vrfutil"
function init()
  vrf:setTickPeriod(5.0)
end

function tick()
  currentSpeed = this:getSpeed()
  if currentSpeed > 3 then
    print("Speed is:", currentSpeed, ". Going too fast.")
  else
    print ("Speed is OK.")
  end
end
```



38. Writing Advanced Scripts with the Behavior Engine

The VR-Forces behavior engine allows you to create behaviors for simulated forces.

38.1. Overview of the VR-Forces Behavior Engine 817
38.1.1 Behavior Trees 817
38.1.2 Unit-Specific Extensions to Standard Behavior Trees 817
38.1.3 Coordination of Subordinate Behaviors 818
38.1.4 Other Behavior Representation Benefits of Behavior Trees 818
38.1.5 What Behavior Trees Are Not 819
38.2. Behavior Node Definition 819
38.2.1 Common Node Attributes 820
38.2.2 Node Execution 821
38.3. Behavior Node Types 822
38.3.1 Task Node 822
38.3.2 Condition Node 822
38.3.3 Sequence Node 823
38.3.4 Loop Node 823
38.3.5 Selector Node 824
38.3.6 Parallel Node 826
38.3.7 Role Node 828
38.3.8 Sync Node 829
38.3.9 Send Node 831
38.3.10 Receive Node 831
38.3.11 Job Node 833
38.3.12 Query Node 833
38.4. Node Member Functions 834
38.5. Behavior Roles 835
38.5.1 Role Definition 835
38.5.2 Role Assignments 837
38.6. Behavior Engine API 837

38.6.1 Declaring a Behavior 837
38.6.2 Using State Variables 838
38.6.3 Getting Role Information 838
38.6.4 Variable Definitions 838
38.6.5 Script Validation 838
38.7. Behavior Integration into Lua Tasks 839
38.7.1 Script Type 839
38.7.2 Behavior Script Components 840
38.7.3 Graphical Display of Behavior 841
38.7.4 Editing with ZeroBrane Studio 843
38.8. Programming Practice for Behaviors 844
38.8.1 Node Names 844
38.8.2 Perform Single Actions in Task Nodes 845
38.8.3 Avoid Excessive Computation in Node Functions 845
38.8.4 Put Task Parameters in State Variables 845
38.8.5 Minimize Global Communication Through State 845
38.8.6 Define Nodes in Variables and Use These Variables for Children of Other Nodes 846
38.8.7 Put Static Options in doIf-Switched Nodes 850
38.8.8 Push Conditions out of Functions and into the Tree 850

38.1. Overview of the VR-Forces Behavior Engine

The VR-Forces behavior engine is a Lua module that provides developers with structures, conventions, and tools to organize and simplify complex behavior scripts. Users are expected to be familiar with VR-Forces tasks and Lua scripting.

The VR-Forces behavior engine was developed primarily to support the creation of unit-level behaviors. VR-Forces already has a capable mechanism for creating entity-level behaviors: you can script tasks in Lua, using an extensible API to the simulation engine and packages such as fsm.lua to organize the flow of control. However, unit behavior is more complex than entity-level behavior in a fundamental way: units have subordinates. While entity behaviors are often composed from simpler entity level behaviors, unit behaviors are composed from simpler unit behaviors and individual subordinate behaviors. The subordinate behaviors typically execute in parallel, with subordinates performing different functions in coordination with one another. The behavior engine helps the developer create complex behaviors with coordinated parallel subordinate behaviors.

38.1.1 Behavior Trees

The VR-Forces behavior engine is based on the concept of behavior trees. Behavior tree engines are common in computer games, simulations, and even robots. Behavior trees provide several benefits to behavior design: they support decomposition of complex behaviors into simpler elements; they provide a visual representation of the decomposition; and they structure the flow of control with standard sub-behavior selection algorithms.

Behavior tree elements are called nodes. Nodes may have one or more children, which are arranged in a tree. The leaves of the tree are primitive actions. Actions in this engine may be VR-Forces tasks, asynchronous jobs, or immediate computations. Interior nodes in the tree may run their children in sequence, loop over their children repeatedly, run them in parallel, or use another standard execution control algorithm. These are all aspects of the VR-Forces behavior engine and common to most behavior tree systems.

38.1.2 Unit-Specific Extensions to Standard Behavior Trees

To support subordinate behaviors, the VR-Forces behavior engine uses extensions to standard behavior trees. "Roles" are parts of the unit behavior that are performed by only part of the unit—a subordinate. Subordinates are assigned to roles based on their type, their "function" in the unit, and other attributes that the developer defines. (Functions are set in the Simulation Object Editor. See "71.4.6 Defining Functions for Subordinates" on page 1410.) A part of the behavior tree reserved for a particular role is identified with a "role node". Multiple subordinates can be allowed to perform the same role, thereby creating multiple "instances" of the role node's sub tree. Typically, a unit behavior contains a parallel node whose children are role nodes, such that the subordinates performing different roles will do them in parallel.

The VR-Forces behavior engine also implements sync nodes. Sync nodes are like barriers that block execution until another sync node is executed elsewhere. These are used for synchronizing control flow in different parallel parts of the tree, particularly in parallel roles being executed by different subordinates.

38.1.3 Coordination of Subordinate Behaviors

Coordination between the tasks performed by subordinates is a key part of designing a unit behavior. Consider a platoon behavior to move to a support position and engage from alternating firing locations. This is a useful stand-alone behavior, and it is also useful as a component of a company (higher echelon) behavior. However, when used in a company behavior, it may be necessary to coordinate an event in the middle of the platoon behavior—for example, the moment the platoon is in position and ready to fire—with actions in the middle of another platoon's behavior. Therefore, the platoon supporting-fire behavior may be slightly different in the context of a company behavior than it is as a standalone platoon behavior.

There are several methods to achieve the coordination necessary between subordinate behaviors.

- First, explicit messages may be sent between the subordinate behaviors and the unit behavior. This approach requires explicit communication actions and a message receiving function. Messages may mimic real-world messages or be "simulation" messages. The latter can be used when in the real world there would be some other form of communication or synchronization. The VR-Forces behavior engine supports this approach with Send and Receive nodes.
- A second approach to coordination is through observation. For example, if a subordinate behavior must wait until another subordinate has completed some action, it may be possible to observe the completion of the action, and so no explicit communication is required. Pilots flying with a wingman can signal intended turns with appropriate motions, thus maintaining radio silence. This approach can be used in VR-Forces behaviors.
- A third approach to coordination is to incorporate the subordinate behavior into the unit behavior, such that the unit behavior has direct access to all of the nodes that control the subordinates. In this approach, coordination can use global variables; this is essentially what the Sync nodes are. This approach does not allow direct reuse of (stand-alone) subordinate behaviors with copying them; however, as noted above, subordinate behaviors may be different in the context of a higher unit behavior. Several behaviors that have been built using the behavior engine are designed in this way.

38.1.4 Other Behavior Representation Benefits of Behavior Trees

Behavior trees are typically used in games and simulations to define the behavior of a single agent. Behavior trees can switch between sub-behaviors when conditions change and the agent's current actions are no longer appropriate. The behavior assigned to an agent can persist indefinitely, switching between different tasks to satisfy one or more goals (such as a mission or self-preservation) over the course of the entire game or exercise. Behaviors created with the VR-Forces behavior engine can produce persistent, intelligent, and robust behavior for both units and individual entities.

38.1.5 What Behavior Trees Are Not

The aim of any behavior model is to describe what an agent should do next in the current situation. In the general case, such as modeling human-level general intelligence, an agent should be able to perform problem-solving processes such as planning, problem recognition, problem-space search, and so on. For many virtual world applications, such as providing an environment of non-player characters for games or training systems, agent behavior can be described adequately with a simple ordering of steps and some simple decision making for branching the order. Behavior developers can effectively use behavior languages and models that abstract and simplify behaviors, even if these models do not support general problem solving.

Behavior trees govern the flow of control in a behavior. The decisions made in the tree are based on computations using information such as input parameters, unit status, locations and state of other entities, characteristics of terrain, and the environment. The behavior tree does not assist directly in any of these computations. Intelligent unit behavior may require terrain analysis, sensor fusion, threat assessment, spatial reasoning, and planning. The behavior engine does not do any of these things. Instead, the behavior engine in VR-Forces allows you to write Lua code and include it in any part of the behavior tree. Therefore, writing behavior is not an exercise in graphically composing a tree from pre-defined blocks, but a process of coding in Lua and organizing the code according to a structured architecture.

38.2. Behavior Node Definition

VR-Forces behaviors that use the behavior engine are Lua scripts that are organized into a behavior tree and executed with the help of the behavior engine Lua module. Behavior trees are composed of nodes, each of which is a chunk of Lua code. The behavior tree receives control every simulation cycle in a call to its tick function. The nodes perform actions and tick their child nodes. The tick functions return a value of success, fail, or running to the parent node. Nodes can contain other Lua functions as well, which perform initialization and exit tasks, start subtasks, check loop conditions, and so on. The tick function runs the Lua functions, selects children and ticks them, and processes results from children.

Because the tick function in each node executes according to a standard definition, the tick function is defined in the behavior engine. Developers do not write tick functions; instead, they indicate the children of a node, provide values for node attributes, and define functions to be called in the tick (if any of these things are relevant for the node). This node description is implemented as a Lua table. This is a simple example of a node description with attributes `nodeType` and `children`. It is a node represented as a Lua table:

```
aNode =
{
  nodeType = "sequence",
  children = {task1, task2}
}
```

where *task1* and *task2* are variables containing node descriptions.

38.2.1 Common Node Attributes

There is a set of node attributes that may be specified for all nodes. These attributes are described below. All of them are optional except for nodeType and sometimes name. All of the attributes that are functions take one argument, which is a node.

- nodeType. A string that defines the type of the node. Required for all nodes. The types available are described in "38.3. Behavior Node Types" on page 822.
- name. A string. This displays in the debugging graph output. (Required for sync nodes—see "38.3.8 Sync Node" on page 829.) It should very briefly describe what the node and the subtree below it do in the behavior.
- doIf. A Lua function. Returns a Boolean. This function is executed after the behavior tree is created and subordinates have been assigned to roles, but before the tree is executed. The doIf function takes a node as an argument; unlike the parameter passed to other functions in nodes, the doIf function is passed its parent node.

  If doIf returns false, then the node is disabled—in effect, it is as if it was never created. The purpose of doIf is not to cause the behavior to perform different actions in different situations that arise during the task; rather it is to make different versions of the behavior in different initial conditions. For example, a behavior might have a subtree related to fire support that is disabled if there is no forward observer, or a behavior might have a subtree of actions related to an optional task parameter that is disabled when the parameter is not present.
- startWhen. A Lua function. Returns a Boolean. The node (tick) is not executed until this function returns true. While it remains false, the node is in a "waiting" state. In this state the node returns "running". The execution of the behavior essentially pauses at the node until the startWhen function returns true.

  A waiting node is treated in a special way by parent parallel nodes: it does not block the completion of the parent (see "38.3.6 Parallel Node" on page 826). If the startWhen never becomes true, the parent node can finish if its other children finish.
- initFn. A Lua function. No return value. This function is executed the first time the node is ticked.
- exitFn. A Lua function. No return value. This function is executed after the node finishes its tick for the last time (and thus has transitioned to a **success** or **fail** state).
- endWhen. A Lua function. Returns a Boolean. This function is executed at the beginning of every tick; if it returns true, then the tick function immediately returns with **success**.
- failWhen. A Lua function. Returns a Boolean. This function is executed at the beginning of every tick; if it returns true, then the tick function immediately returns with **fail**. This function is run before endWhen.

38.2.2 Node Execution

The following is a pseudocode description of the tick function of a generic node:

```
  node.state = "created" -- Set when node is created
  ...
  function tick(node)
     result = node.state
     if node.state is in {"created", "success", "fail"} then
         node.initFn(node)
         node.state = "waiting"
     end
     if node.state == "waiting" then
         if node.startWhen(node) then
             node.state = "running"
         else
             result = "running"
         end
     end
     if node.state == "running" then
         if node.failWhen(node) then
             node.state = "fail"
             result = "fail"
         else if node.endWhen(node) then
             node.state = "success"
             result = "success"
         end
         Process node according to type (see processNode function
  pseudocode for different nodes in section 3 below), ticking children
  as necessary. Set node.state.
         result = node.state
     end
     if result is in {"success", "fail"} then
         node.exitFn(node)
     end
     return result
  end
```

Some notes on the pseudocode:

- After a node finishes (enters a success or fail state), if it is ticked again it will reinitialize. This can happen if the node is in a loop in the behavior tree.
- If startWhen is not defined, it is treated as if it returns true.
- If endWhen or failWhen is not defined, it is treated as if it returns false.

The above algorithm happens in one tick.

38.3. Behavior Node Types

This section covers the node types for use when scripting with the behavior engine.

38.3.1 Task Node

A task node is a leaf node in the behavior tree that runs VR-Forces tasks and performs other computations. Its nodeType is "task". It has one unique attribute:

- task. (required) A Lua function. Takes a node as an argument. Returns an integer. Return value meaning:
  - -1 - "fail"
  - 0 or nil - "success"
  - Positive - A VR-Forces task ID

If the task function returns nil, 0 or -1, the node tick function immediately sets the node state and exits.

Normally, the task function starts a VRF task and returns the VRF task ID. For example:

```
task = function(node)
  Compute some parameter values
  return vrf:startSubtask("...
  end
```

When the task function starts a task, the node tick function returns "running" (unless the startSubtask call fails). Each subsequent tick, the tick function checks the task's status and sets the node.state to "running", "success", or "fail" depending on the status of the task.

You can use the task function to call `vrf:executeSetData()` (or `vrf:sendSetData()`). A `setData` function executes immediately, so the task function should return 0 or nothing.

You can also use the task function simply to compute something. Computations might produce data for the condition function in a subsequent condition node or parameters for a task node, for example. The results of the computation would be saved in state variables (see below) so that the subsequent node can access them. The task function would typically return nothing or 0.

38.3.2 Condition Node

A condition node is a leaf node with nodeType "condition". It has one unique attribute:

- condition. (required) A Lua function. Takes a node as an argument. Returns true or false.

If the condition function returns true, the node tick function sets the node state to **success**. Otherwise, it returns **fail**.

38.3.3 Sequence Node

A sequence node is a node that executes child nodes in sequence. Its nodeType is "sequence". It has one attribute beyond the common node attributes:

- children. (required) A list of node descriptions; that is, a Lua table whose elements are table descriptions of Lua nodes.

The tick function of a sequence node executes the tick functions of the nodes in its "children" list one at a time until they all succeed, or one returns **fail**. If one returns **running**, then the sequence node returns **running**. The children are executed in the order that they appear in the children list.

The sequence node tick function acts as follows:

```
node.index = 1 -- Set when node is created
...
function node.processNode()
  status = "success"
  while status == "success" and
    node.children[node.index] != nil do

    status = node.children[node.index].tick()
    node.index = node.index + 1
  end
  node.state = status
end
```

38.3.4 Loop Node

A loop node repeats its child until a condition fails or the child fails. The nodeType is "loop". The node has two attributes beyond the common attributes; both attributes are required:

- doWhile. (required) A function returning a Boolean. The node exits when this function becomes false.
- child. (required) A node description (table).

The loop node processNode function simply ticks the child and then tests doWhile. The algorithm is:

```
function node.processNode()
  status = node.child.tick()
  if status == "fail" then
    node.state = "fail"
  elseif status == "success" and doWhile == false then
    node.state = "success"
else
    node.state = "running"
  end
end
```

The loop node returns **running** while its child is running. If the child returns **fail**, the loop returns **fail** immediately. The doWhile function is only checked when the child has finished and returns **success**.

The second time through a loop, the child starts in the **success** state; however, when a node in **success** state is ticked, it acts as if it is being ticked for the first time. Thus, the initFn is called again (see "38.2.2 Node Execution" on page 821).

38.3.5 Selector Node

A selector node is a node that attempts to execute its children until one succeeds or all fail. The nodeType is "selector". It has two attributes beyond the common attributes:

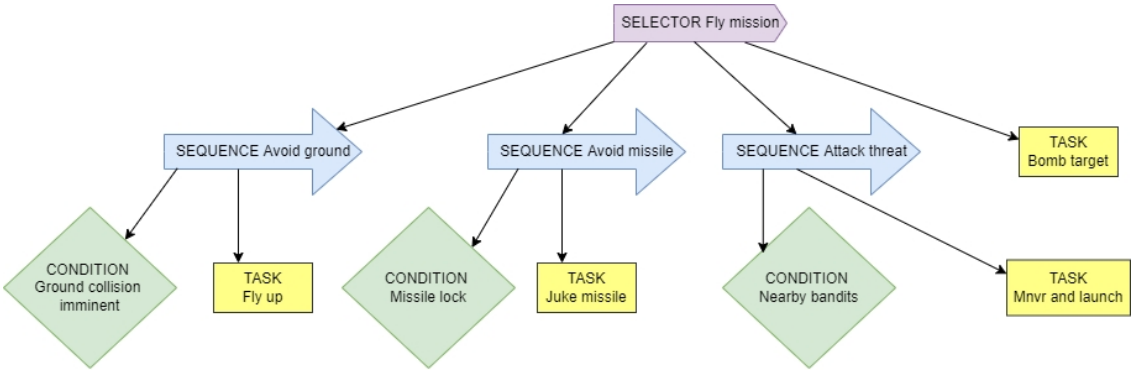
- mode. (optional) A string that is either "reactive" or "fallback". The default mode is "reactive".
- children. (required) A table of node descriptions.

The node has two modes of operation: reactive and fallback. These are similar algorithmically, but are appropriate for completely different applications.

Reactive Mode

In reactive mode, each tick the selector node ticks its first child node (the order of the child nodes is determined by the order of nodes in the "children" list). If the node returns **fail**, then the selector ticks the next node. This repeats until one child returns **success** or **running**; the selector node then returns the same result. If all nodes return **fail**, then the selector returns **fail**. Typically, the children are each sequences consisting of a condition followed by an action or subtree; the effect of the selector node in reactive mode is to select the highest-priority child node whose conditions are satisfied. This selection is made every tick, making the process very reactive. This design pattern is illustrated in Figure 283. Each tick, the selector node checks all of the nodes from left to right. When the condition for a node is true, the corresponding task is run. When none of the conditions are true and the Bomb Target task is finally complete, the selector node exits.

Figure 283. Example selector node in reactive mode



The pseudocode for the reactive mode selector tick function is:

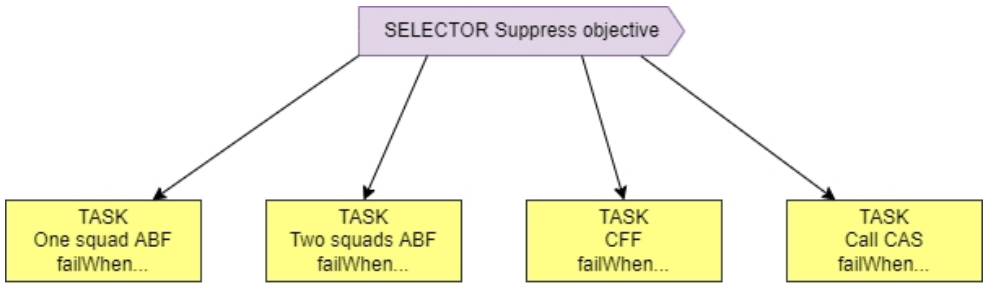
```
node.index = 1 -- Set when node is created
...
function node.processNode()
  for childIndex = 1 to node.children.size() do
    status = node.children[childIndex].tick()
    if status is in {"success", "running"} then
      if node.index > childIndex then
        node.children[childIndex].stopNode()
      node.index = childIndex
      break
    end
  end
  node.state = status
end
```

The `stopNode()` function in the code above stops VR-Forces tasks and jobs and cancel queries that are in a node or its children.

Fallback Mode

In fallback mode, the children are again ticked until one of them returns **success** or **running** or all return **fail**. However, in each tick, the children are ticked beginning with the one that was **running** the last tick. The children of the selector node in this mode are not prioritized reactions, but methods for accomplishing an objective in order of preference. When the selector ticks a child, only if it fails does the selector try the next child in the list. The children are, in effect, fallback methods for one another. Figure 284 illustrates this design pattern. Each of the first children has a failWhen attribute that causes the node to return **fail** if the method does not successfully suppress the objective:

Figure 284. Example selector node in fallback mode



The pseudocode for the fallback mode selector tick function is:

```
node.index = 1 -- Set when node is created
...
function node.processNode()
  status = "fail"
  for childIndex = node.index to node.children.size() do
    status = node.children[childIndex].tick()
    if status is in {"success", "running"} then
      if node.index > childIndex then
```

```
          node.children[childIndex].stopNode()
        node.index = childIndex
        break
      end
    end
    node.state = status
  end
```

38.3.6 Parallel Node

A parallel node is a node that ticks all its children every tick, in parallel. The nodeType is "parallel". There are three attributes beyond the common set:

- children. (required) A table of node descriptions.
- successPolicy. (optional) A string with the value "one" or "all"; the default is "one". This attribute specifies how many children must return **success** for the parallel node to return **success**.
- letAllFinish. (optional) A Boolean; the default is true. The return value of a parallel node may be determined before all children have returned **success** or **fail**; if this attribute is true, the parallel node continues to return **running** until all of the children have finished. For example, if a parallel node has a successPolicy of "one" and one child returns **success**, the value of the parallel node is **success**, regardless of whether the other children have finished or are still running. Setting this attribute to true makes the parallel node return **running** until all children have finished.

When a child node finishes, but the parallel node has not finished, the parallel node does not continue to tick that child.

If a parallel node has letAllFinish = false, and a child node result allows the parallel node to determine what its result is (for example, successPolicy = "one" and a node returns **success**), the parallel node will finish and return. However, if any of its children are still running, they will be stopped. Stopping a task means that the engine calls a function like `vrf:stopTask()`, stopping a job involves `job:cancel()`, and so on.

If a child node's startWhen function is false, it is not considered in the letAllFinish test. That is, if letAllFinish is true, and startWhen is false in the remaining running children, then the parallel node will finish (status changes from **running** to **success** or **fail**).

The processing of a parallel node in the tick function depends on the values of its attributes. The different cases are listed below.

successPolicy = "one", letAllFinish = true

```
node.running = node.children -- Set when node is created
node.result = "fail" -- Save the result until all finish
...
function node.processNode()
  for child in node.running do
    status = child.tick()
    if status is in {"success", "fail"} then
      Remove child from node.running
    end
    if status == "success" then
```

```
      node.result = "success"
    end
  end
  if node.running is empty then
    node.status = node.result
  else
    node.status = "running"
  end
end
```

successPolicy = "all", letAllFinish = true

```
node.running = node.children -- Set when node is created
node.result = "success" -- Save the result until all finish
...
function node.processNode()
  for child in node.running do
    status = child.tick()
    if status is in {"success", "fail"} then
      Remove child from node.running
    end
    if status == "fail" then
      node.result = "fail"
    end
  end
  if node.running is empty then
    node.status = node.result
  else
    node.status = "running"
  end
end
```

successPolicy = "one", letAllFinish = false

```
node.running = node.children -- Set when node is created
...
function node.processNode()
  for child in node.running do
    status = child.tick()
    if status is in {"success", "fail"} then
      Remove child from node.running
    end
    if status == "success" then
      node.status = "success"
      break
    end
  end
```

```
  if node.running is empty and
      node.status == "running"
      -- Finished all children without a success

      node.status = "fail"
  end
end
```

successPolicy = "all", letAllFinish = false

```
node.running = node.children -- Set when node is created
...
function node.processNode()
  for child in node.running do
    status = child.tick()
    if status is in {"success:, "fail"} then
      Remove child from node.running
    end
    if status == "fail" then
      node.status = "fail"
      break
    end
  end
  if node.running is empty and
    node.status == "running"
    -- Finished all children without a fail

    node.status = "success"
  end
end
```

38.3.7 Role Node

A role node is a node that defines a portion of the behavior that will be performed by one or more subordinates of the unit performing the behavior. A unit behavior will often be composed of multiple roles such as "support", "assault" and "security" working together. Each of the component parts of the behavior is a child of a different role node.

The role node has nodeType "role". A role node has two required attributes:

- role. A string; the name of the role. This must match the roles defined for the behavior (see Behavior Roles below).
- child. A node description (table).

Several subordinates of the unit performing the behavior may be assigned to the same role (see Behavior Roles below). When this is the case, the behavior engine creates an instance of the child node for each of these subordinates. The role node executes each of these in parallel, much like a parallel node. The success policy of the role node is "one", and the let-all-finish policy is true.

If no subordinates are assigned to the role, the role node returns "success". (If in the behavior design it is desirable to ignore the role node when there are no subordinates assigned to the role, this condition can be tested in a doIf function in the role node.)

Roles may be marked as "critical" in the behavior; see "38.5.1 Role Definition" on page 835. When a role is critical, it must have at least one subordinate assigned to it, otherwise it fails. If the role node fails for this reason or because all of the instances of the child node fail, then the whole behavior fails.

Each instance subtree beneath a role node has its own private environment for saving state data. State data is described in "38.6.2 Using State Variables" on page 838 and "38.4. Node Member Functions" on page 834.

38.3.8 Sync Node

A sync node is a signaling feature for synchronizing control flow in parallel branches of the subtree. This mechanism involves a global state variable for recording the states of the sync nodes. The behavior engine manages the global variable and the state changes of the nodes.

The nodeType of a sync node is "sync". There are no special attributes for the sync node. In fact, none of the common attributes for nodes are allowed except for doIf. Thus initFn, startWhen, endWhen, failWhen, and exitFn are not allowed. The name attribute is required for a sync node. All sync nodes in the tree that are synchronized with each other must have the same name.

When control flow in the tree reaches a sync node, it is essentially blocked until control flow reaches all identically named sync nodes in all other parts of the tree. When a sync node is ticked, the behavior engine checks all other identically named sync nodes. If they have not been ticked, then the first node returns **running**. If all other nodes have been ticked, then the sync node returns **success**. All of the other nodes return **success** the next time they are ticked as well.

Sometimes a behavior subtree with sync nodes must be executed multiple times in a loop. After a sync node returns **success**, if it is ticked again it acts as if it is being ticked for the first time.

Sync Node Execution

Figure 285 is a diagram of the states that a sync node goes through. The node starts in the "inactive" state before it is ticked for the first time. When it is ticked, it enters the "waiting" state and returns **running** unless all other nodes with the same name are already in the "waiting" state; in that case, the node remains in the inactive state and returns **success**. The engine also changes the states of all other nodes with the same name to "free". A node in the "free" state becomes "inactive" and returns **success** the next time it is ticked. All of the nodes start "inactive" and change to "waiting" when they are ticked, until only one more node is inactive. When the last node is ticked, it remains "inactive" (to be ready for another synchronization event) but sets all the other nodes to "free". "Free" nodes change to "inactive" when they are ticked, and return **success**.

Figure 285. Sync node states. References to other nodes (such as "other waiting nodes") indicate other nodes with the same name.

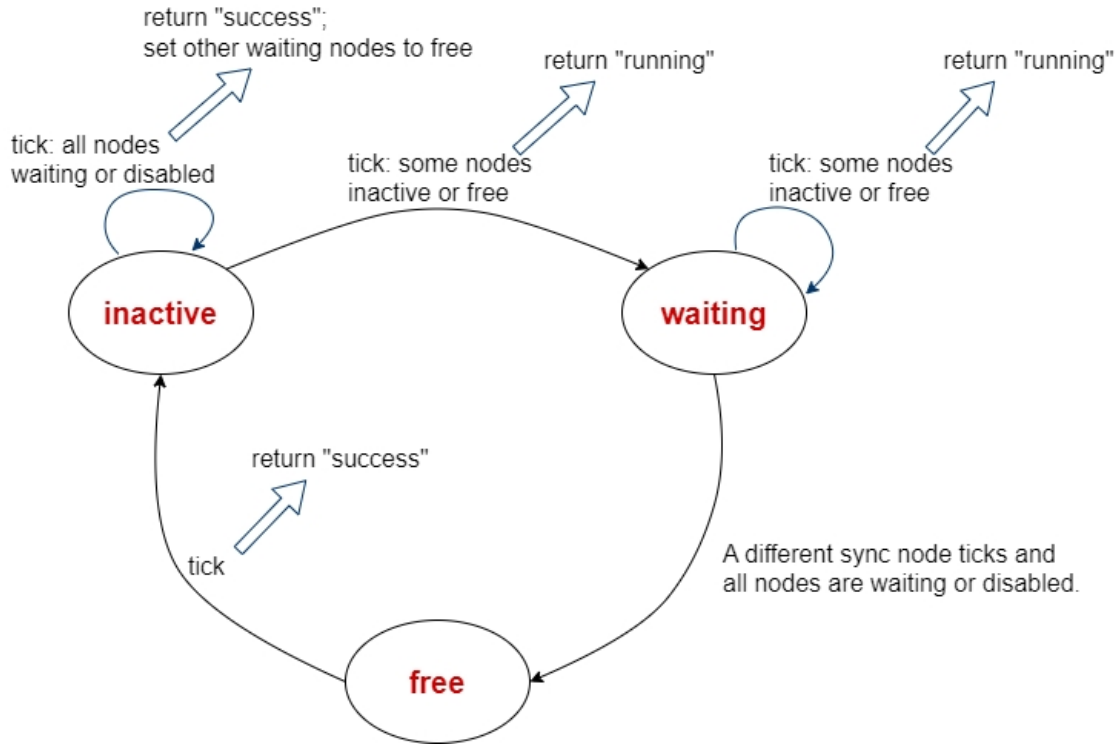
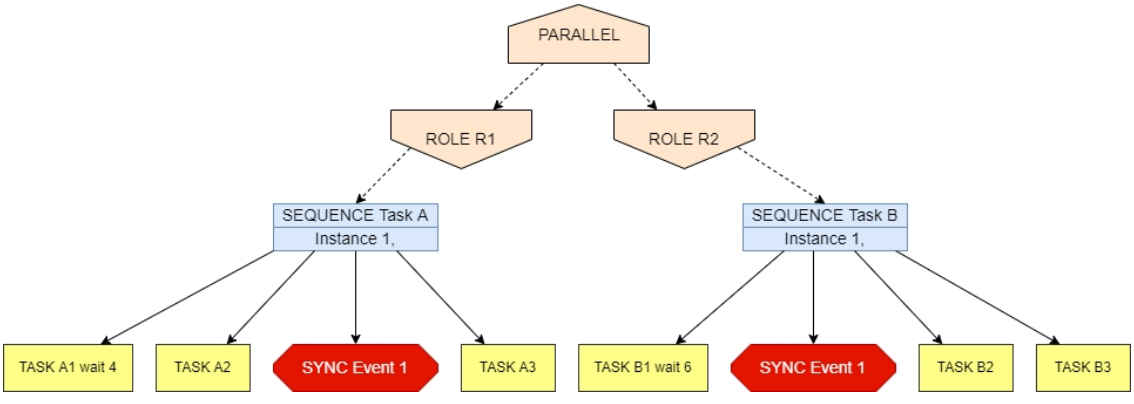


Figure 286 shows a simple application of sync nodes. In this example, task A3 in role R1 is intended to start at the same time as task B2 in role R2. The sequence in Task A is blocked with sync node Event 1 returning **running** until sequence Task B ticks sync node Event 1 (or vice versa).

Figure 286. Example of using sync nodes to coordinate the start of tasks A3 and B2



"38.8.6 Define Nodes in Variables and Use These Variables for Children of Other Nodes" on page 846 presents the code represented by this tree.

Disabled Sync Nodes

There is an additional sync node state, "disabled". Sync nodes become disabled in certain conditions when a part of the behavior tree is not active (below). When a node is disabled, it is ignored for synchronization purposes; it is as if it is always in the "waiting" state. If, however, a disabled node is ticked again, it acts as if it was in the "inactive" state, going to the "waiting" state.

Sync nodes are disabled by the engine to attempt to prevent a behavior from blocking unintentionally when something goes wrong. Nodes are disabled if:

- A doIf function evaluates to false in a node, all sync nodes in the subtree under that node are disabled.
- A child of a sequence node returns **fail**, thus causing the sequence node to fail. Sync nodes are disabled in the subtrees of all children to the right of the failed child.
- A parallel node finishes while some of its children are still running (that is, letAllFinish = false), then the running children are stopped, and sync nodes in the subtrees under the children are disabled.

38.3.9 Send Node

A send node is used for sending a message to another object. This message may be text, or it may be a structured message (Lua table). In effect, this node wraps the `vrf:sendTextMessage` and `vrf:sendObjectMessage` functions. It is intended to support "generic-table" object messages by default, as these are expected to be the basis for domain messages such as SITREPS.

The node type of a send node is "send". A send node has several attributes beyond the common attributes:

- messageType. (optional) A string that defines the VR-Forces message type to send. This is the name of an object message type, or it can be "text" to send an unstructured text message. If omitted, the message type is "generic-table".
- sendTo. This attribute specifies the recipient of the message. It can either be a simObject, or the string "unit" or "subordinates". If the value is "unit", the message is sent to this object's superior and all of the superior's subordinates (peer entities). If the value is "subordinates", then the message is sent to all of this object's subordinates.
- messageFn. A Lua function. The function takes on argument: a node. The function must return either a string (for text messages) or a table.

A send node finishes with a status of **success** the first time it is ticked.

38.3.10 Receive Node

A receive node provides a means for a behavior to receive and process messages. It is a wrapper for the script entry points `receiveTextMessage` and `receiveObjectMessage`, and the function `vrf:registerForObjectMessage`. It makes use of the msgUtil Lua library to manage message queues.

The node type of a receive node is "receive". A receive node has the following attributes beyond the common attributes:

- messageType. (optional) A string defining the type of messages to receive. This is the name of an object message type, or the strings "text" or "all". If the value is "all", all types of object messages and text messages are received. Otherwise, only the given object messages are received. If omitted, the type is assumed to be "generic-table".
- sender. (optional) This attribute specifies who the sender of the message must be. This attribute may be a simObject, or it may be the strings "unit" or "subordinates". The meanings of these two strings are the same as in the sendTo attribute in Send nodes. If no sender is specified, then messages from all senders are received.
- acceptFn. A Lua function. This function takes one argument, a node. This function is called on each message received. The function should return either true, false, or nil. A value of true causes the node to finish with status **success**. A value of false causes the node to finish with status **fail**. A value of nil allows the node to keep running to process additional messages.
- timeOut. (optional) A number. If specified, timeOut indicates the maximum number of seconds that the node can remain running without receiving a message. If the timeOut is exceeded, then the node returns with status **fail**.

Accessing the Message

The acceptFn takes the node as its argument. A special node function `node.result()` provides access to the message received. This function returns a table with the fields:

- messageType. A string defining the type of the message. This is an object message type, or "text".
- message. The message body, which is either a string or a table.
- sender. A simObject that is the sender of the message.

Message Registration

Normally, when a script wishes to receive object messages, it must call `vrf:registerForObjectMessage` and specify the type of message to receive. However, when a receive node specifies a message type. the behavior engine makes the `registerForObjectMessage` call. The behavior script does not need to make this call.

If the message type is specified to be "all", the behavior engine does not call `registerForObjectMessage` for every possible object message type. In this case, the behavior script must make these calls for the messages of interest.

Asynchronous Entry Points

Since the behavior engine uses the msgUtil package, the receiveTextMessage and receiveObjectMessage entry point functions are already defined, and they should not be defined in the behavior script. For more information, see the Lua documentation for msgUtil.

Message Queues

The behavior engine runs the behavior tree synchronously, which is at regular intervals in a tick function. Behavior trees therefore process messages synchronously as well. The behavior engine makes use of the message queues in msgUtil to store incoming messages between ticks, and then receive nodes process them in order when they are ticked.

During each tick, the first (oldest) message is taken from the node's queue and the node's acceptFn is called. If the acceptFn returns true or false, the node's status is changed and it finishes processing. If the acceptFn returns nil, then the behavior engine gets the next message from the node's queue and again calls acceptFn. The queue is processed until there are no more messages or acceptFn returns true or false.

38.3.11 Job Node

A job node is similar to a task node in that it starts an activity in VR-Forces that completes in a later tick. A job node is used to start an asynchronous job. The nodeType is "job". This node has one attribute beyond the common attributes:

- job. A function that returns an AsyncJob object (defined in the VR-Forces Lua API).

The job function returns the result of a Lua API function that creates an AsyncJob. For example, `vrf:findCoverPoint(...)`. This node will return "running" until the asyncJob has finished. The job node should use the exitFn to retrieve the results of the job; the results can be saved in a state variable for use by other nodes. The exitFn uses a special node `function node.result()` which will return the result of the asyncJob. The form of this results depends on which Lua API function was invoked in the job function.

This is an example of a job node:

```
{
  nodeType = "job",
  name = "Get positions",
  job = function(node)
    return vrf: findCoverPoint(…)
    end,
  exitFn = function(node)
    setStateVar("coverPoints", node.result())
    end
}
```

38.3.12 Query Node

A query node allows some user interaction in the behavior, to the extent that user interaction is supported in Lua scripts. The nodeType is "query". A query node has two attributes beyond the common ones:

- query. This is a table with the following keyword elements:
  - title. A string that will display as the title of the dialog box.
  - message. A string that will display in the dialog box to prompt the user.
  - options. A table whose items are strings. These strings are the responses from which the user can select when responding to the query.
  - displayStyle. An integer specifying how the options will display:
    - 0 - Set of buttons
    - 1 - Radio buttons
    - 2 - Combo box
- timeOut. A number. (optional) The number of seconds that the behavior will wait for the user to respond to the query. If there is no response in this time, the query is canceled.

The options in the query attribute are only the arguments to the `vrf:askUserQuestion()` function.

The query node returns **running** when it is waiting for a response, and always returns **success** when it completes. The result of the choice can be found by calling the `node.result()` function (in the exitFn). Result() returns an integer corresponding to the user choice, with 0 as the first choice.

38.4. Node Member Functions

Each of the node attributes (described in "38.2.1 Common Node Attributes" on page 820 and "38.3. Behavior Node Types" on page 822) that is a function has one argument: a node. These functions are doIf, startWhen, initFn, endWhen, failWhen, exitFn, task, condition, doWhile, and job. The node argument can provide context in the function. A node object provides several functions to provide this context:

- subordinate(). Returns a simObject. This is the subordinate that is executing the behavior node. This value is nil for the unit running the behavior from the top; however, underneath each role node, a subordinate will be assigned to run the subtree.
- instanceN(). Returns an integer >= 1. If more than one subordinate has been assigned to a role, then each has a unique index assigned as well. This index will correspond to the order in which the subordinate has been assigned to the role (see role assignment, below). This index can be used to cause the behavior to be slightly different for each subordinate in a role; for example, by positioning the subordinates instanceN() x spacing along a line.
- setRoleStateVar(variableName, value). Each instance of each role has a private environment in which to store state variables (in addition to the global environment; see "38.6.2 Using State Variables" on page 838). This function is used to store variables in the private state. variableName is a string identifying the variable; value is any Lua value (a string, number, object defined in the Lua API, or table).

Note that if this function is called from a place in the behavior that is not beneath a role node, the variable will be stored in the global environment.

- getRoleStateVar(variableName). Returns the value of the state variable with variableName (which is a string).
- result(). Job and query nodes only. For jobs, returns the result of the job. For queries, returns an integer corresponding to the user choice, with 0 corresponding to the first choice.
- status(). The state of the function; a string ("created", "waiting", "running", "success", "fail"). Normally the status of nodes should not be needed in a behavior, except perhaps for printing debugging output. Status is used by the engine for implementing node logic.

38.5. Behavior Roles

Behaviors may include multiple roles in their definition. A role is a part of the unit behavior that is executed by a subordinate of the unit. Typically, multiple roles will be executed in parallel. Roles have no meaning for behaviors that are run on individual entities.

There are two parts to a role definition. The first part is a definition of the role name and specification of which subordinates can be assigned to each role. This part is written in the form of a table with keywords "types", "isCritical", "maxInstances", and "scoreFn". These are explained in "38.5.1 Role Definition" below. The second part is comprised of one or more role nodes in the behavior tree that have a "role" attribute that matches the role name in part one. Role nodes are described in "38.3.7 Role Node" on page 828.

If you write a behavior for an entity without subordinates, then the role definition table may be omitted from the behavior.

38.5.1 Role Definition

The role specification is given in a table in the script. The elements of the table are:

- name. (required) A string. This identifies the role. The "role" attribute of a role node must match this name exactly.
- types. (optional) A list (table) of entity type strings. Only subordinates with these types can be assigned to the role. Type strings may use "-1" in a field to indicate that any value matches. If the types attribute is missing, any type of subordinate is allowed to have the role.
- isCritical. (optional) Boolean. If this attribute is true, then the role is critical: at least one subordinate must be assigned to the role, and at least one instance of the role must return "success" for the behavior to succeed. If no subordinates can be assigned to the role, the behavior fails before starting. If a critical role node returns "fail", the entire behavior will fail at the end of the tick. If this element is not included in the definition table, "false" is assumed.

- maxInstances. (optional) An integer. The maximum number of subordinates that can be assigned to the role. If this element is not included in the definition table, the value is assumed to be 1.
- scoreFn. (optional) A function that takes a simObject and an integer as arguments and returns a number. The simObject passed is a subordinate, and the integer is an index into the list of subordinates currently in the unit. This function provides a score for a possible assignment of the simObject to the given role. Higher scoring subordinate-role pairs will be assigned before lower scoring ones. A score of 0 means that the subordinate should be assigned to the role only if it cannot be assigned to any other role. A score of < 0 means that the subordinate will never be assigned to the role. If the scoreFn is missing from the definition table, a score of 0 is assumed.

This is an example of a role specification:

```
myRoles =
{
  {
    name = "first maneuver section",
    types = {"1:1:-1:1:-1:-1:-1", "1:1:-1:2:-1:-1:-1"},
    maxInstances = 2,
    isCritical = true,
    scoreFn = function(sub, index)
      local fn = subordinateFunction(sub)
      if fn == "PL" or fn == "PLWM" then
        return 1.0
      else
        return 0.0
      end
  },
  {
    name = "second maneuver section",
    types = {"1:1:-1:1:-1:-1:-1", "1:1:-1:2:-1:-1:-1"},
    maxInstances = 2,
    isCritical = false,
    scoreFn = function(sub, index)
      local fn = subordinateFunction(sub)
      if fn == "PSGT" or fn == "PSWM" then
        return 1.0
      else
        return 0.0
      end
  }
}
```

In this example, we assume that there is a function subordinateFunction(sub) that returns the function of the simObject sub in the unit. These scoreFns assign up to two vehicles to each of two maneuver sections, with preferences for certain roles in each section.

Score functions can use many factors to identify subordinates for a role. They can:

- Get the object type and look up parameters.
- Compute distances to control objects.
- Use index positions in the subordinate list.
- And so on.

38.5.2 Role Assignments

Subordinates are assigned to roles in the init() function of the script. The role assignments are permanent for the duration of the behavior. Each subordinate can only be assigned to one role. It is possible for a subordinate not to be assigned to any role.

Optimal assignment of subordinates to roles is a very complex problem in general, so the behavior engine does not try to find an optimal assignment. The assignment algorithm works as follows:

1. Compute the score for each subordinate for each role.
2. Go through the critical roles in list order and assign one subordinate to each; assign the highest scoring subordinate available. (If any critical role does not have a subordinate assigned, exit with failure.)
3. Go through the non-critical roles in list order and assign one subordinate to each; assign the highest scoring subordinate available, but do not assign anyone if the highest score is 0.
4. Find the highest score for any role among all remaining subordinates and make that assignment. If there are multiple scores with the same value, prefer an assignment to a role with fewer subordinates already assigned. Repeat until all subordinates have been assigned or all roles have been filled to maxInstances.

Throughout the process, constraints are maintained: assignments must satisfy the type constraint, roles can only have maxInstances subordinates assigned, and scores cannot be < 0.

38.6. Behavior Engine API

The behavior engine does not have a large API that scripts use to define or run behavior trees; most of the information is in the behavior node elements described in the previous sections. This section describes the function necessary to register a behavior tree, and the other functions provided by the behavior engine.

38.6.1 Declaring a Behavior

After the behavior and roles have been defined as tables, the tree is registered with the engine with the function:

```
function defineCommand(params)
```

where *params* is a table with two fields:

- topNode. (required) The table defining the behavior tree.
- subordinateRoles. (optional) The table defining the roles.

For example:

```
defineCommand({topNode = behavior,
  subordinateRoles = roles})
```

In this example, the behavior tree is a table assigned to variable behavior, and the roles are in variable roles.

38.6.2 Using State Variables

The Lua functions in the behavior nodes (as well as Lua code outside the nodes) can save data into a global environment. (In many behavior tree implementations, this environment is called a "blackboard".) Global state can be used to pass information between nodes of a tree (although this reduces the modularity of the tree) and for saving state for a scenario save and reload. In behaviors using the behavior engine, state is saved and accessed using two functions:

- setStateVar(name, value). Where name is a string, the name of the state variable, and value is any value (including tables). Does not return a value.
- getStateVar(name). Where name is a string, the name of the state variable. Returns the value.

Inside the engine, the checkpointState global variable is used to store the behavior state; however, this variable should not be accessed directly in the script. The behavior engine takes care of setting the checkpoint state mode.

38.6.3 Getting Role Information

There are several functions in the behavior engine API related to subordinates and their assignments to roles.

- roleForSubordinate(subordinate). Argument subordinate is a simObject. This function returns a string which is the name of the role assigned to the subordinate.
- subordinatesAssignedToRole(name). Argument name is a string, the name of a role. This function returns a list (table) of simObjects that are assigned to the given role.
- isRoleCritical(name). Argument name is a string, the name of a role. This function returns a Boolean indicating whether the given role is critical (that is, the isCritical element in the definition is true).

38.6.4 Variable Definitions

The behavior engine defines the following global variables for use in the behavior script:

- _subordinates. This is a list (table) of simObjects, which are the subordinates of the unit. These are the direct subordinates, not individuals in the unit. The list contains only capable subordinates that are capable of moving and shooting.

38.6.5 Script Validation

Although this is not part of the behavior engine API, script validation is another function that is performed by the engine. When VR-Forces loads the script, the engine reads the description tables provided in the **defineCommand** function and checks the script for

errors. Each node description is checked to make sure that all required attributes are present, and that all attributes present are valid attributes. It also checks the type of each attribute provided. If there are any errors, the node name and a description of the error are printed to the Warn console.

38.7. Behavior Integration into Lua Tasks

The behavior engine is a set of Lua files that are used in the normal VR-Forces Lua scripting mechanism. Because a behavior is, underneath, only a Lua script, all of the Lua scripting features apply:

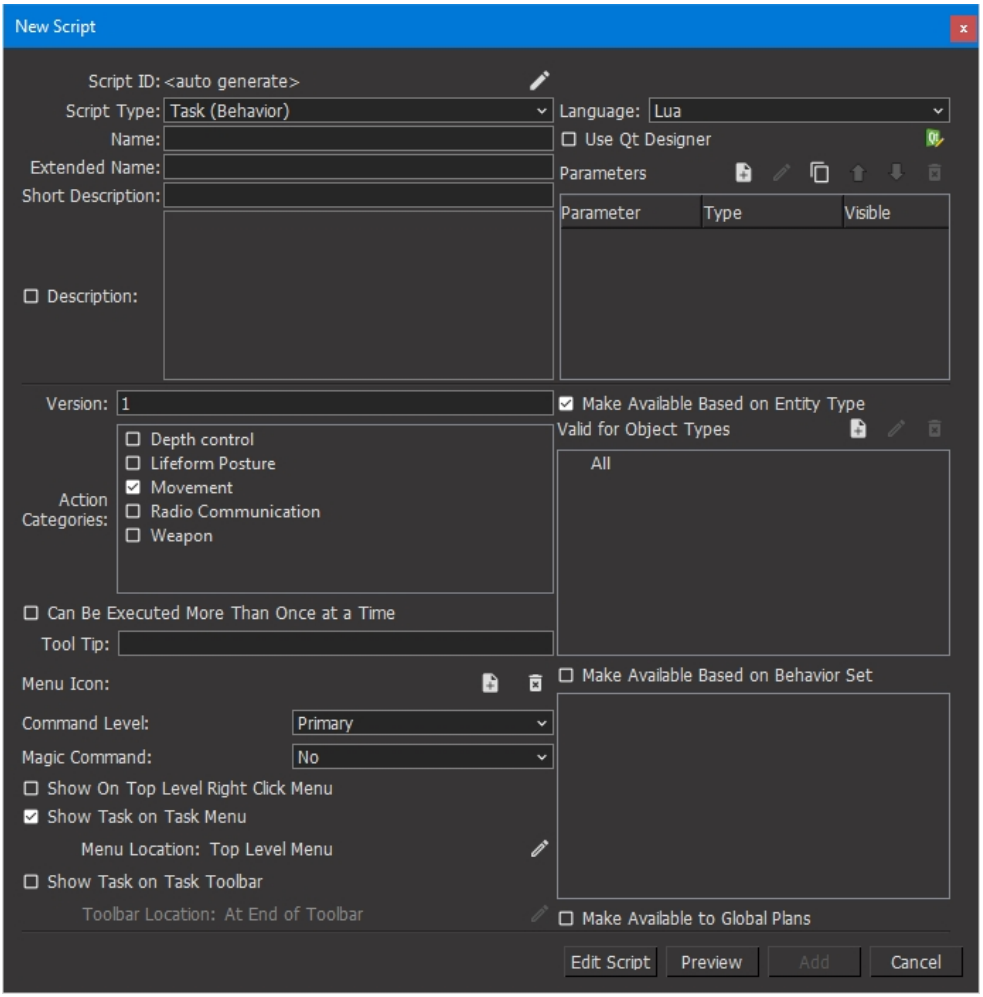
- It is created from the Script Manager dialog
- It uses script metadata, including name, script parameters, menu configuration, applicable entity types, and so forth.
- It is managed by the entity's Task Manager; it can be part of a plan.
- And so on.

This section describes the ways in which behaviors are different from normal Lua tasks.

38.7.1 Script Type

When creating a behavior that will use the behavior engine, select the "Task (Behavior)" script type in the New Script dialog box (Figure 287).

Figure 287. New Script dialog box



When you select this option, the template for the task is tailored to a behavior engine behavior instead of a normal task script.

The New Script dialog sets the category of the task to "movement" by default. Most unit behaviors will involve movement. However, unit behaviors do many things, and it is more likely that multiple behaviors need to run on the unit at the same time, perhaps to task different subordinates of the unit. For this reason, it is often desirable to clear the movement box.

38.7.2 Behavior Script Components

A behavior tree script should normally start with comments describing what the behavior does, followed by two require statements:

```
require "vrfutil"
require "behaviorEngine"
```

Next, a behavior script typically has definitions of global functions, a table of role definitions, and tables of node definitions. Finally, there is a call to `defineCommand()`.

The behavior script should not include definitions of `init()`, `tick()`, `saveState()`, or `loadState()`. The behavior script should not call `vrf:setCheckpointMode()`.

Figure 288 on the next page is an example of a complete behavior script that prints a "hello" from each of the unit subordinates.

38.7.3 Graphical Display of Behavior

The behavior engine includes an experimental feature that displays the behavior tree graphically. This feature uses an installation of parts of Graphviz—in particular, the dot layout program—in the *./bin64/Graphviz* directory.

> The behavior tree is currently supported only on Windows.

The behavior engine writes a file with the name "be_<unit name><behavior name>.gv" into the *./bin64* directory, then runs dot on that file to produce a file "be_<unit name><behavior name>_buffer.html". The engine further creates a file "be_<unit name><behavior name>.html", which contains JavaScript that displays the *..._buffer.html* file and refreshes it twice per second. The behavior engine starts an instance of the Microsoft Edge browser to execute this JavaScript code. When a behavior is run on a unit and the unit's Notify level is set to Debug, the behavior engine opens an Edge browser and displays the behavior. The behavior state is updated whenever the state changes, but no more often than every 0.5 second.

In the graphical display, node types are distinguished by shapes and color. The outline of a node indicates its status:

- gray - not started
- blue - running
- dashed green - success
- dashed red - fail

Graph edges are dashed to indicate they are parallel to another branch—that is, they come out of a parallel or role node). Edges are bold blue if they go into a running node; thus the current locus of control can be seen in the sequence of bold blue edges that goes from the top node of the tree to a leaf.

Inside the node graphics, the engine displays the nodeType, the node name, and indicators for the presence of doIf, startWhen, endWhen, and failWhen functions. Parallel nodes have indicators when the successPolicy is "one" or lefAllFinish is "false"; selector nodes include an indicator if the mode is "fallback"—that is, values other than the default.

Task parameters and state variables are displayed in tables in the graphical output. If a node is an instance under a role node, then its private environment is displayed in the node. The values of the variables in all these cases are only displayed if they are scalars or locations; also, only the first item in a table is displayed.

Figure 289 on page 843 is an example of the graphical output.

Figure 288. Example of Complete Behavior Script

```
-- Hello task

require "vrfutil"
require "behaviorEngine"

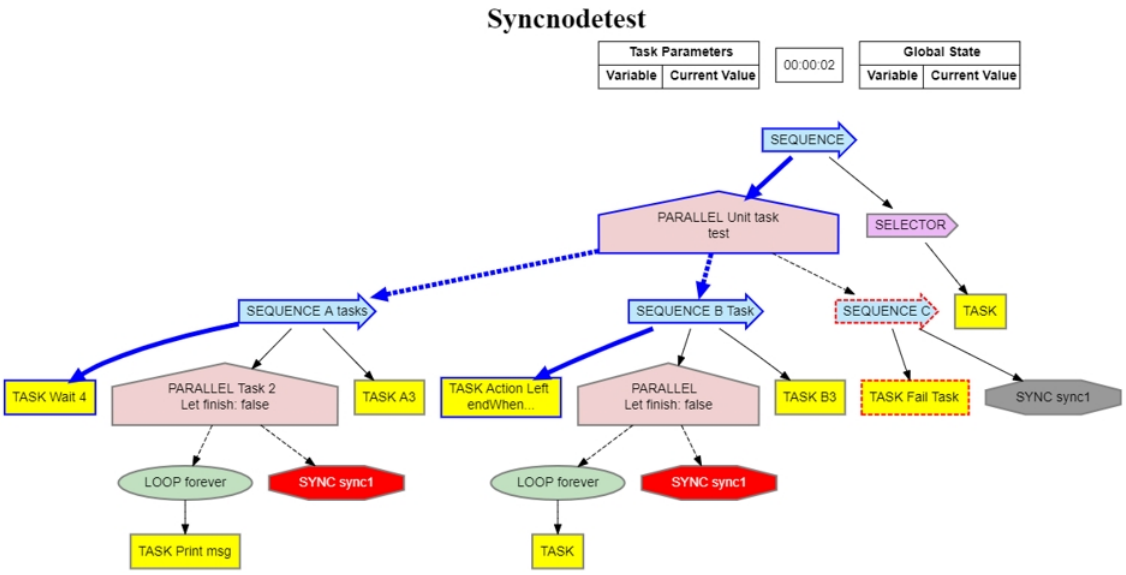
roles =
{
   {
      name = "All subs"
      ,maxInstances = #_subordinates
   }
}

-- Define behavior nodes.
ind_beh =
{
  nodeType = "task",
     name = “Print sub. name”,
     task = function(node)
     printWarn("Hello from ",
       node.subordinate():getName())
     return 0
  end
}

behavior =
{
   nodeType = "role",
   role = "All subs",
   child = ind_beh
}

-- Register the roles and top node with the behavior engine.
defineCommand{
  subordinateRoles = roles,
  topNode = behavior
}
```

Figure 289. Example of a graphical display behavior tree during execution



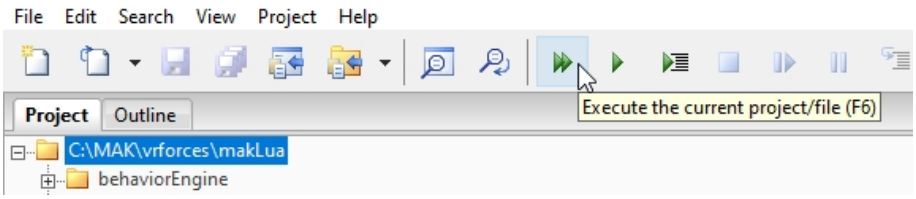
38.7.4 Editing with ZeroBrane Studio

The ZeroBrane Lua IDE can be used to validate the behavior syntax and preview a behavior tree graphically before the behavior is run in VR-Forces.

➤ **To use ZeroBrane to validate the behavior syntax and preview the behavior tree:**

1. Open a Lua script from the Script Manager dialog.
2. Click the Execute button. This executes the script using the Lua interpreter built into the ZeroBrane IDE. The interpreter reports any Lua syntax errors in the script.

Figure 290. Execute the script



3. If there are references to VR-Forces-defined tables in the global part of the script that will be executed when the script is loaded, these will cause errors because ZeroBrane does not have the VR-Forces definitions loaded. For example, vrf and this are nil. Typically, in a behavior, functions and tables are only defined, not run, when the behavior script is loaded, so not many of these nil references will be executed. If some of them are run, they can be de-activated for the purposes of checking the script with the IDE with this code:

```
-- Causes error:
-- location = this:getLocation3D()
-- Safe for checking script in editor:
if this ~= nil then
```

```
    location = this:getLocation3D()
end
```

If there are no Lua syntax errors in the script, and no references to undefined tables or functions, then the behavior engine checks the validity of the script (missing or incorrect node elements, and so on). If you do not have another installation of ZeroBrane, Lua, or Graphviz, the behavior engine creates the files to show the tree graphically, and launches an Edge instance (on Windows) to display it. The graphics files are saved in the project directory: *./userData/scripts*.

You can use the ZeroBrane editor to edit behavior scripts (or any Lua scripts) even without running VR-Forces. Simply select a file in the Windows File Explorer and open it using ZeroBrane, edit the script as desired, and then execute it in ZeroBrane. The Lua engine in ZeroBrane checks the Lua code for errors. Next, the behavior engine checks the node definitions for validity, and then a graph is generated.

38.8. Programming Practice for Behaviors

The behavior engine design for VR-Forces deliberately includes Lua code in the node definitions. This design decision means that a standard programming language is available for tackling what is often the hardest part of building behaviors—geometric calculations, situation awareness, threat assessment, and so forth—while a specialized engine handles flow of control.

The incorporation of general Lua code also means that behavior developers can write incomprehensible behaviors. On the other hand, if developers follow some guidelines and try to use the behavior engine for what it is intended, then the behaviors will be much easier to develop and maintain. In particular, the developers (and others who review the behavior) can understand much of the behavior just from the graphical display.

This section presents guidelines to help you write better behaviors.

38.8.1 Node Names

Node names display with every node in the graphical display. Names are not used for anything else, except in the case of sync nodes. Therefore, behavior developers should create node names that provide clear information to someone viewing the graph.

These are some guidelines for node names:

- Use multiple words to say very concisely what the node, or subtree under the node, is doing.
- Do not include the type of the node, because that is automatically printed. For example, task node names should not include "task".
- If a task node starts a VR-Forces task, use the task name.
- If the node includes a condition such as doIf, startWhen, endWhen, failWhen, or doWhile, try to include a brief description of the condition.

When the node names are read and displayed in the browser, they can be interpreted as HTML; therefore, do not use any special characters such as angle brackets, quotes, or anything else that might be interpreted.

38.8.2 Perform Single Actions in Task Nodes

The task function in a task node can be used to compute parameters, update state variables, issue Set commands, and issue subTask commands. In fact, all of these operations could be performed in the same task function. To make the behavior tree more visually understandable, however, different actions should be performed in different nodes rather than putting them all in one task function.

It is easy to test a condition in the task function and return -1 or 0 depending on the result. Instead of doing this, use a Condition node to make it clear that the purpose of the action is to test a condition.

38.8.3 Avoid Excessive Computation in Node Functions

All of the node functions—doIf, initFn, startWhen, task, and so on—are general Lua functions, and they may contain any amount of code. Normal coding practices should be applied: if a function is too large, factor it into pieces. Lower-level functions can be defined at the global level in the behavior script and referenced in the node function body. Predicate functions (returning a Boolean) should not in general cause side effects. (One exception to this is doWhile, which may need to increment a counting variable or something as part of the test, since there is no counting mechanism built into the node definition.) If the work accomplished by the computations is too much to concisely describe in the node name, then consider breaking the computations apart into multiple nodes.

38.8.4 Put Task Parameters in State Variables

The arguments to VR-Forces tasks that are started in task nodes can be computed directly in those nodes, if they are fairly simple. However, it may be useful to save copies of the arguments in state variables to ensure that they are shown in the graphical output.

38.8.5 Minimize Global Communication Through State

The Lua functions in the behavior tree nodes can set and test state variables. State variables can be used to communicate information between nodes. There are two reasons that this is undesirable to do this:

- The connection between nodes is hidden in the graph display, and difficult to see in the script itself. This makes the behavior more difficult to understand.
- One potential benefit of using behavior trees to represent behavior is that they can be very modular. For example, it is possible to add nodes to a sequence or parallel node without affecting the other child nodes. However, if subtrees have hidden connections in the form of state variable dependencies, some of the modularity is lost.

Uses of state variables that minimize these problems include:

- State variables that are set and used within a node. For example, a loop counter that is set and used in a loop node.
- State variables that pass information as locally as possible. For example, computing some information in a job node or a task node, and then using that information in the next node in a sequence.
- State variables that are derived from task parameters and other static information, that are computed in the first initFn, and that are not changed further in the script.

38.8.6 Define Nodes in Variables and Use These Variables for Children of Other Nodes

A behavior tree description is a nested table. This description can be written in the script with nested curly braces. However, this approach to writing scripts leads to long table definitions, which is poor practice (similar to writing long pages of procedural code). For example, the simple tree in Figure 286 on page 830 looks like this when written out.

```
top =
{
  nodeType = "parallel",
  children =
  {
    {
      nodeType = "role",
      role = "R1",
      child =
      {
        nodeType = "sequence",
        name = "Task A",
        children =
        {
          {
            nodeType = "task",
            name = "A1 wait 4",
            task = function (node)
              return
                vrf:sendTask(
                  node.subordinate(),
                "wait-duration",
                {seconds_to_wait = 4})
              end
          }
          ,{
            nodeType = "task",
            name = "A2",
            task = function (node)
                     return 0 end
```

```
        }
        ,{
          nodeType = "sync",
          name = "Event 1"
        }
        ,{
          nodeType = "task",
          name = "A3",
          task = function (node)
                   return 0 end
        }
      }
    }
  }
,{
  nodeType = "role",
  role = "R2",
  child =
  {
    nodeType = "sequence",
    name = "Task B",
    children =
    {
      {
        nodeType = "task",
        name = "B1 wait 6",
        task = function (node)
          return
            vrf:sendTask(
              node.subordinate(),
            "wait-duration",
            {seconds_to_wait = 6})
          end
      }
      ,{
        nodeType = "sync",
        name = "Event 1"
      }
      ,{
        nodeType = "task",
        name = "B2",
        task = function (node)
                 return 0 end
      }
      ,{
        nodeType = "task",
        name = "B3",
        task = function (node)
                 return 0 end
```

```
          }
        }
      }
    }
  }
}
```

When written using variables to represent each node, all of the nesting is unnecessary, and the code looks like this:

```
sync1 =
{
   nodeType = "sync",
   name = "Event 1"
}
A1 =
{
   nodeType = "task",
   name = "A1 wait 4",
   task = function (node)
      return vrf:sendTask(
         node.subordinate(),
         "wait-duration",
         {seconds_to_wait = 4})
      end
}
A2 =
{
   nodeType = "task",
   name = "A2",
   task = function (node) return 0 end
}
A3 =
{
   nodeType = "task",
   name = "A3",
   task = function (node) return 0 end
}
SeqA = {
   nodeType = "sequence",
   name = "Task A",
   children = {A1, A2, sync1, A3}
}
R1 =
{
   nodeType = "role",
   role = "R1",
   child = SeqA
```

```
}

B1 =
{
   nodeType = "task",
   name = "B1 wait 6",
   task = function (node)
      return vrf:sendTask(
         node.subordinate(),
         "wait-duration",
         {seconds_to_wait = 6})
      end
}
B2 =
{
   nodeType = "task",
   name = "B2",
   task = function (node) return 0 end
}
B3 =
{
   nodeType = "task",
   name = "B3",
   task = function (node) return 0 end
}
SeqB =
{
   nodeType = "sequence",
   name = "Task B",
   children = {B1, sync1, B2, B3}
}
R2 =
{
   nodeType = "role",
   role = "R2",
   child = SeqB
}

top =
{
   nodeType = "parallel",
   children = {R1, R2}
}
```

The behavior illustrated here is simplistic, but it also shows opportunity for code reuse that can be used in more complex behaviors. Note that the sync node description is exactly the same in two places in the tree, so it is only described once. Other nodes are

highly similar; for similar nodes that are repeated in the tree, a creator function could be defined. For example, the task nodes A1 and B1 are the same except for their name and wait time. They could be defined this way:

```
function createWaitTask(name, time)
    local desc = {}
    desc.nodeType = “task”
    desc.name = name
    desc.task = function (node)
      return vrf:sendTask(node.subordinate(),
         "wait-duration", {seconds_to_wait = time})
      end
    return desc
end
A1 = createWaitTask(“A1 Wait 4”, 4)
B1 = createWaitTask(“B1 Wait 6”, 6)
```

38.8.7 Put Static Options in doIf-Switched Nodes

Make use of doIf functions to disable parts of the tree that are not relevant for the initial conditions. The doIf condition can test anything that is set before the start of the behavior: taskParameters, role assignments, instance numbers in roles, the unit type, unit or subordinate parameters, subordinate function, or anything derived from this type of information. As a behavior developer, you should avoid using condition nodes to select behavior branches at run time when the condition information is available at the beginning of the behavior; such selections make the runtime structure of the behavior needlessly complex. Instead, use doIf functions near the top of the tree to select behavior branches. With this approach, each branch handles one behavior condition and should be much simpler to design.

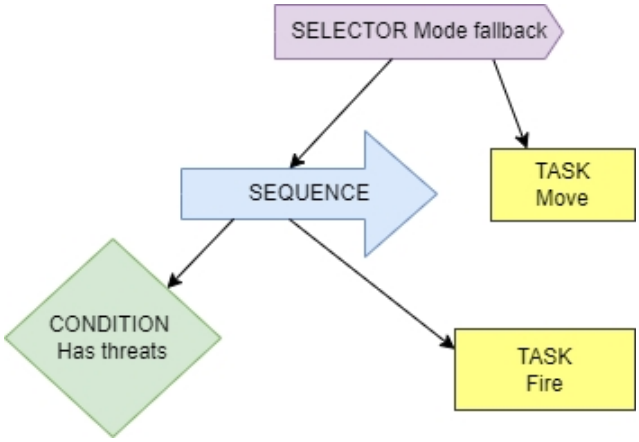
38.8.8 Push Conditions out of Functions and into the Tree

All flow of control in a behavior should ideally be defined in the behavior tree architecture. Although it is possible to put a complex, branching flow of control in the Lua code in the nodes, this hides the flow from the graph view and makes the behavior harder to understand. For example, a task node might have a task function written like this:

```
task = function(node)
  if getStateVar("threats") then
    return vrf:sendSubtask(node.subordinate(), "fire...
  else
    return vrf:sendSubtask(node.subordinate(),"move...
  end
end
```

This key branch in the logic should be implemented as a selector node in fallback mode, as shown in Figure 291.

Figure 291. Explicit branching on conditions in the behavior tree





39. Using Qt Designer for Script Dialog Boxes

You can let VR-Forces create dialog boxes for scripted tasks and sets using default layout rules or you can design the dialog boxes yourself using Qt Designer.

39.1. Introduction to Using Qt Designer for Scripts 854
39.2. Selecting Qt Designer for Dialog Box Layout 854
39.3. Adding Parameters When Using Qt Designer 854
39.3.1 Specifying Choices for the Choice Parameters in Qt Designer 857
39.3.2 Specifying Categories for a Simulation Object in Qt Designer 858
39.4. Adding Design Elements 859

39.1. Introduction to Using Qt Designer for Scripts

When you create a scripted task or set that requires input from a user, VR-Forces can automatically build a dialog box with fields for each parameter. These dialog boxes are built using a default layout scheme. This layout scheme is functional, but it may not always capture relationships between different parameters, particularly dependencies. Also, the resulting dialog boxes may not be as esthetically pleasing as well designed dialog boxes should be. As an alternative to automatic dialog box creation, you can use Qt Designer to lay out dialog boxes. This capability is integrated into the script creation process.

The VR-Forces graphical user interface is built using Qt, a cross-platform GUI SDK. Qt Designer is a utility that developers can use to quickly lay out GUI design elements. It is beyond the scope of this manual to fully explain how to use Qt Designer. This chapter provides enough information about it to help you start designing with the utility.

39.2. Selecting Qt Designer for Dialog Box Layout

By default, VR-Forces automatically builds dialog boxes for scripted tasks and sets.

➤ To specify use of Qt Designer for dialog box layout, in the New Script or Edit Script dialog box, select the Use Qt Designer check box.

39.3. Adding Parameters When Using Qt Designer

When you create or edit a script, you can add parameters in the New Script or Edit Script dialog box. If you select Qt Designer for dialog box layout, any parameters that you have already defined in the New Script dialog box are implemented in Qt Designer using the default dialog box layout (which you can change). However, once you select Qt Designer, you cannot add any more parameters in the New Script dialog box. You can only add parameters in Qt Designer.

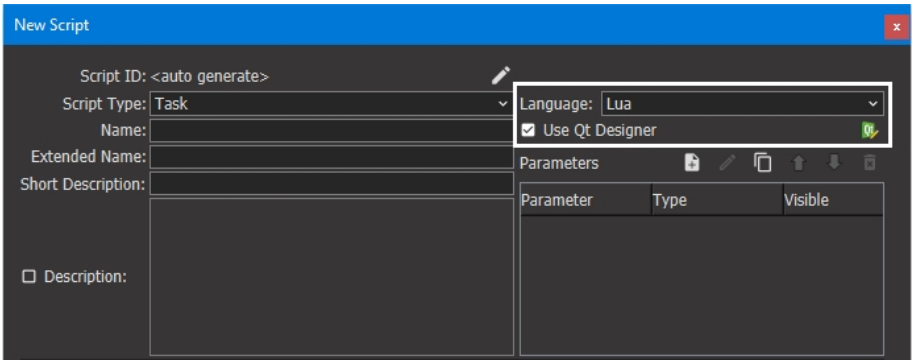
If you decide to use Qt Designer before you add parameters in the New Script dialog box, then all parameters are added using Qt Designer.

> ⚠ If you lay out a dialog box using Qt Designer and then unselect the Use Qt Designer check box, all of the layout information that you added in Designer is lost, except that if you rearranged the order of parameters, those changes affect the order in which the parameters are listed in the Edit Script dialog box. Any parameters that you added in Designer are still valid and part of the script.

➤ **To add parameters in Qt Designer:**

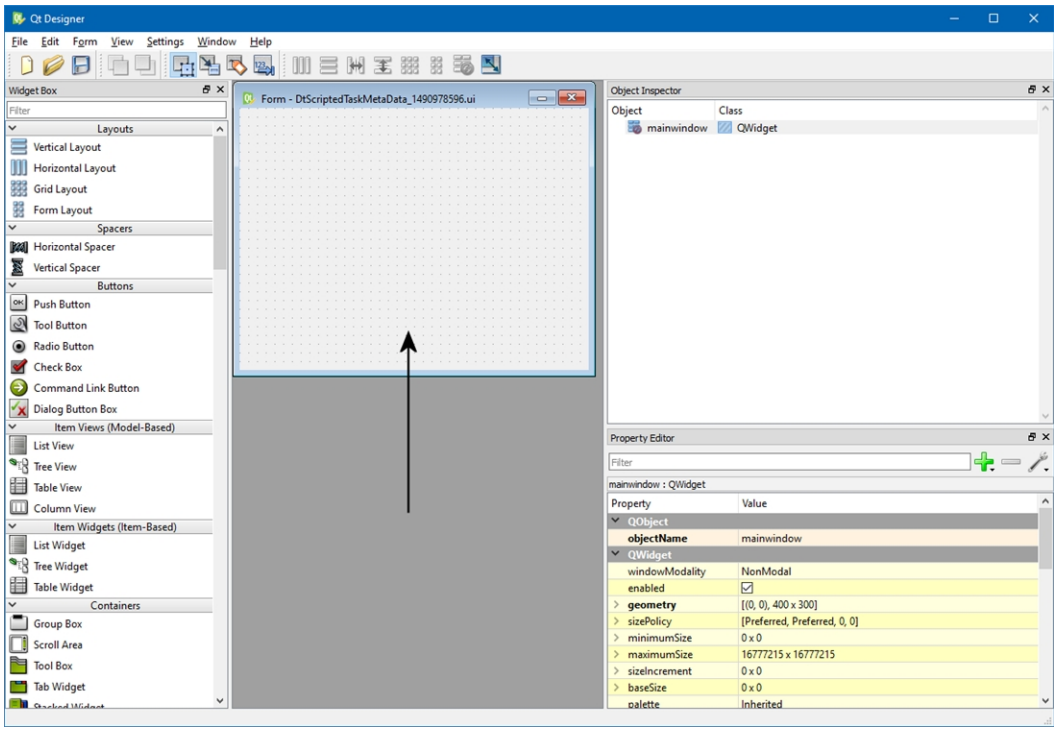
1. In the New Script or Edit Script dialog box, select the Use Qt Designer check box (Figure 292).

Figure 292. New Script dialog box



2. Click the Qt button ( ) to the right of the Qt Designer check box. Qt Designer opens (Figure 293). If the script already has parameters, it displays a default dialog box layout for the script. If there are no parameters, it displays a blank canvas.

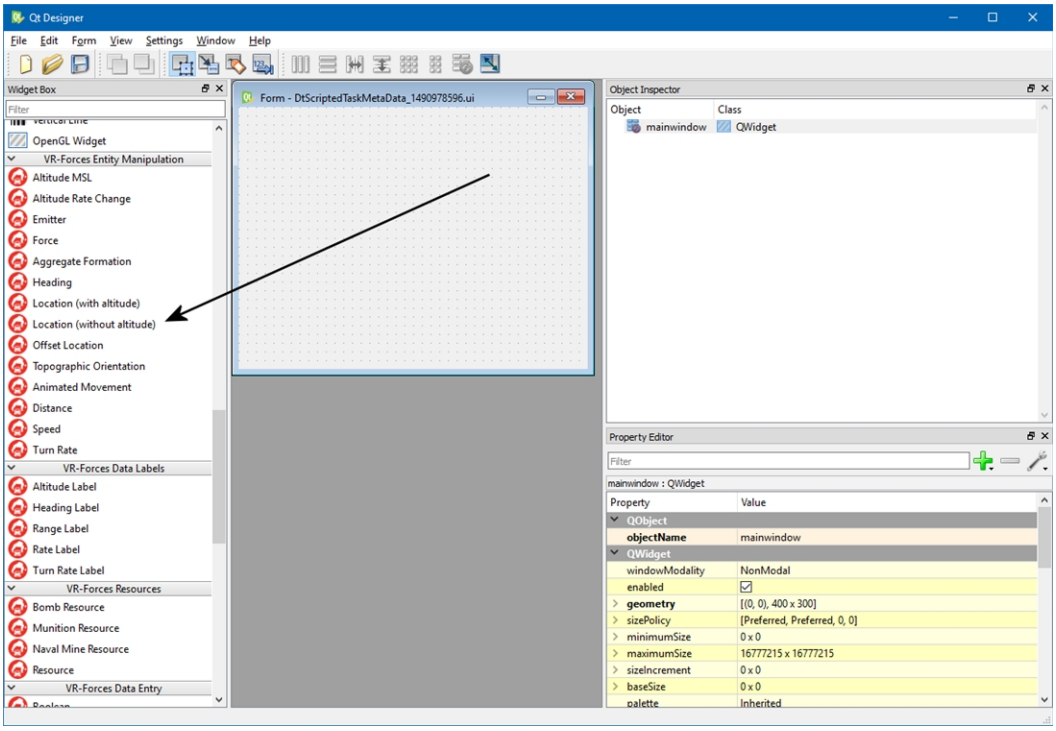
Figure 293. Qt Designer with blank canvas



> You must use the canvas that is created when you open Designer. If you close it and open a new one from the File menu, it might not work.

3. Scroll through the widget box until you see the list of widgets for VR-Forces script parameters (Figure 294).

Figure 294. VR-Forces widgets



4. Drag a parameter onto the dialog box canvas.
5. In the Property Editor, specify the required and optional information for the parameter type, such as a label, tooltip, or default value.

   All of the parameters have these properties:

   - **objectName**. The name to be used by the Lua script for this parameter. For example, if you give a parameter the name **taskDestination**, in the Lua code for the script, the parameter will be accessed as **taskParameters.taskDestination**.
   - **label**. This is the text used on the dialog box that is generated for the task. If you leave this field blank, the parameter name is used.
   - **toolTip**. Optional text for a tool tip.
   - **indent**. The number of pixels to indent the parameter on the task dialog box. This allows you to apply some formatting to the dialog box.
   - **isVisible**. Specifies that the parameter be included on the script dialog box. (You might want a parameter to be available in the code, but not have it set by a user when the script is assigned.)

   Many parameters let you specify a default value or state.

   Some parameters let you specify a range for the acceptable values:

   - **rangeBottom**. The lowest acceptable value for this parameter.
   - **rangeTop**. The highest acceptable value for this parameter.

The Boolean parameter type lets you create a check box. You can specify that the default be checked or unchecked.

6. Add design elements such as dividers and labels as desired.

> Nothing in Qt Designer will prevent you from adding non-VR-Forces properties. However, you cannot use non-VR-Forces properties in scripts.

7. Choose a layout option on the Form menu. Doing so preserves your layout. If you do not select a layout option, VR-Forces might rearrange the layout when it displays the dialog box for the script.
8. In Qt Designer, choose **File > Save** or click the Save button. If this is a system script, Qt Designer saves a UI file with the layout information. If this is a scenario script, the layout information is saved in the scenario.
9. Close Qt Designer.

39.3.1 Specifying Choices for the Choice Parameters in Qt Designer

The Choice parameters require a list of choices, which are added to the dialog box as option buttons or as a list. You can add choices, edit them, and delete them.

Option buttons have a property called **buttonGroup**. Buttons within the same button group are mutually exclusive. If you add more than one group of option buttons to a script, you must edit the **buttonGroup** property to make sure that the different groups have a different value for the property.

➤ **To add a drop-down list of choices:**

1. From the VR-Forces Data Entry section, drag the Choice (Drop-Down List) widget onto the canvas.
2. In the Property Editor, click the Value column for the **values** property. The text changes to Change String List.
3. Click Change String List. The Dialog - Qt Designer dialog box opens. Add a string for each choice you want to make available:
    a. Click New. A text box is added to the list.
    b. Type the string you want to be displayed.
    c. Repeat this procedure for each string you want to be in the drop-down list.
    d. Click OK. The dialog box closes. The strings you entered are listed in the Value column for the values property.

➤ **To add option button choices:**

1. From the VR-Forces Data Entry section, drag the Choice (Multiple Option Buttons) widget onto the canvas. It creates a widget with one button that has the text Radio Button.
2. In the Property Editor, click the Value column for the **values** property. The text changes to Change String List.

3. Click Change String List. The Dialog - Qt Designer dialog box opens. Add a string for each choice you want to make available:
    a. Double-click the text Radio Button to make it editable.
    b. Change the text to something meaningful for this parameter.
    c. Click New. A text box is added to the list.
    d. Type the string you want to be displayed for the next option button.
    e. Repeat steps c and d for each option button you want.
    f. Click OK. The dialog box closes. The strings you entered are listed on the canvas and in the Value column for the values property.

39.3.2 Specifying Categories for a Simulation Object in Qt Designer

The Simulation Object parameter lets you build a list of categories on which to filter the display of simulation object types in a dialog box.

➤ **To add a category to the simulation object list:**

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.
2. Click the Add button () on the Filters line. The Choose Filter dialog box opens.
3. Select a filter from the Filters list.
4. Click OK.
5. Repeat the procedure to add as many filters as you want.
6. Click OK.

Editing the Category List for a Simulation Object Parameter

You can edit the list of filter categories for a Simulation Object parameter. The edit operation is essentially a replacement of the current category with a new one. You can achieve the same effect by deleting the current category and adding a new one.

➤ **To edit the list of filter categories in a Simulation Object parameter:**

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.
2. In the window, select the filter that you want to edit.
3. Click the Edit button () on the Filters line. The Choose Filter dialog box opens.
4. Select the new filter category.
5. Click OK. The original category is removed and the new one is added.
6. Click OK.

➤ **To delete a category from a Simulation Object parameter:**

1. Double-click the Filter widget on the form. A dialog box opens. The top line reads Filters and it has several icons.

2. In the window, select the filter that you want to delete.
3. Click the Delete button (). The category is removed from the list.
4. Click OK.

39.4. Adding Design Elements

You can add design elements such as dividers, group boxes, and indentation. You do so by dragging the appropriate widgets from the list onto the canvas and setting their properties. Please see the Qt Designer documentation for details about how to design dialog boxes.



VII. Set Data Requests

Set data requests affect the state or behavior of a simulation object, such as its speed or posture. They take effect immediately. Assignment of a set data request does not override the task that a simulation object is executing. By contrast, if you assign a simulation object a task, it stops its current task and begins the new task. You can send set data requests from a plan or from the Set menu.

You can create new set data requests using the same Lua scripting process that you can use to create scripted tasks.

Some set data requests only apply to entity-level scenarios. Others only apply to aggregate-level scenarios. These set data requests indicate the applicable simulation type with an entity ( ) or unit ( ) icon in the margin.

The Set menu is context sensitive. It only shows the options that apply to the selected simulation object.

The graphical user interface does not necessarily know about the state of a simulation object. When you open a dialog box for a Set command, any data in the dialog box represents either default values or the most recently used value. Do not assume that a dialog box shows current values.

> You can check the current status of most simulation object state data in the Information dialog box.

You can filter simulation object selection lists. For details, see "29.1.9 Filtering the Object Selection Lists" on page 565.

The instructions for setting simulation object state direct you to use options on the Set menu. However, all the options available on the main menu are also available on context-sensitive menus. A limited number of set data requests are available on the Sets Toolbar and on the Last Selected Object Panel.

You can also set the state of tactical graphics. For details, see "Chapter 48. Setting the State of Tactical Graphics" on page 1037.



40. Setting the State of Entity-Level Simulation Objects

This chapter explains how to set simulation object states for entities and units in entity-level scenarios. Some of the set data requests also apply to aggregate-level scenarios. They are cross-referenced from "Chapter 41. Setting the State of Aggregate-Level Simulation Objects" on page 913.

> You can also use the Entity Control Panel to set an individual entity's state. See "29.2.2 Using the Buttons in the Entity Control Panel" on page 569.

Set data requests for aerodromes and runways are covered in "67.3. Aerodrome and Runway Set Data Requests" on page 1295.

40.1. Active Sonar Mode 866
40.2. Add Jamming Target 866
40.3. Autonomous Actions Enabled 866
40.4. AIS Data 867
40.5. Aiming 868
40.6. Aircraft Lighting 868
40.7. Altitude 869
40.8. Appearance 869
40.9. Armed 870
40.10. Articulated Part 870
40.11. Assume Unknown is Hostile 870
40.12. Attached 871
    40.12.1 Detaching a Tactical Graphic 871
40.13. Attention Grabbing 871
40.14. Automatically Close Doors 872
40.15. Base 872
40.16. Callsign 873
40.17. Capabilities 873
40.18. Collision Avoidance Enabled 873

40.19. Concealed ... 873
40.20. Countermeasures Auto Launch ... 874
40.21. Current Speed ... 874
40.22. Damage Percentage ... 875
40.23. Destroyed ... 875
40.24. Detonation Fuse Type ... 875
40.25. DI-Guy Characteristics (Appearance, Head, Body Weight) ... 876
40.26. DI-Guy Hand Item ... 876
40.27. Disembarked ... 877
40.28. Embarked ... 877
40.29. Emergency Beacon ... 878
40.30. Emitter ... 878
40.31. Engagement Zones ... 879
40.32. Force ... 879
40.33. Formation ... 879
40.34. Freeze Fuel Levels ... 880
40.35. Freeze Movement ... 880
40.36. Freeze Munition Levels ... 881
40.37. GPS Jammer ... 881
40.38. Ground Clamp Fidelity ... 881
40.39. Heading ... 882
40.40. Hide Occupants ... 882
40.41. IFF ... 883
40.42. Invisible ... 884
40.43. Invulnerable ... 884
40.44. Jammer State ... 885
40.45. Jamming Mode ... 885
40.46. Laser Code ... 885
40.47. Laser Power and Wavelength ... 886
40.48. Loadout ... 886
40.49. Location ... 887
40.50. Look At ... 888
40.51. Markings ... 888
40.52. Name Tag ... 889
40.53. Navigation Enabled ... 889
40.54. Navigation Preferences ... 890
40.55. Notify Level ... 890
40.56. Orbit ... 890
40.57. Ordered Speed ... 891
40.58. Pitch ... 892
40.59. Posture (Human) ... 892

40.60. Preferred Trajectory Height ... 893
40.61. Promote From Clutter ... 893
40.62. Radar Mode ... 894
40.63. Rank Patch ... 894
40.64. Remote Sensor ... 895
40.65. Remove Jamming Target ... 895
40.66. Reorganize ... 895
40.67. Resources ... 896
40.68. Restore ... 896
40.69. Rules of Engagement ... 896
40.69.1 How Fire-When-Fired-Upon Works ... 897
40.70. Sector Blanking ... 897
40.71. Sector of Responsibility ... 898
40.72. Sensor Aim ... 899
40.73. Sensor Field of View Color ... 899
40.74. Sensor Zoom ... 899
40.75. Skill Level ... 900
40.76. Sonar Depth ... 900
40.77. Spot Reports ... 901
40.78. Stow Hand Item ... 902
40.79. Subsystem Enabled ... 902
40.80. Superior ... 902
40.81. Surface Vessel Lighting ... 903
40.82. Surrendered ... 903
40.83. Synchronize Laser Code ... 903
40.84. TACAN Transmitter ... 904
40.85. Tanker Lighting ... 904
40.86. Target ... 905
40.87. Target Priority List - Add ... 905
40.87.1 Viewing the Target Priority List ... 905
40.88. Target Priority List - Clear ... 906
40.89. Target Priority List - Update ... 906
40.90. Tasked by Superior ... 906
40.91. TDL Messaging ... 907
40.92. Tracer Use ... 907
40.93. Undetectable ... 907
40.94. Unit Patches ... 908
40.95. Urgency (Human) ... 908
40.96. Set Data Requests for Units ... 909
40.96.1 Assign Transport ... 909
40.96.2 Dismounted ... 909

40.96.3 Dismounted Headquarters Section 909
40.96.4 Lock Embarkation 910
40.96.5 Mounted 910
40.96.6 Mounted Crew 910
40.96.7 Mounted Headquarters Section 910
40.96.8 Ordered Formation 911
40.96.9 Unloaded 911

40.1. Active Sonar Mode

If an entity has an active sonar system, you can set the sonar mode.

- RPR FOM 1.0 does not support active sonar.
- To disable the active sonar, use the Subsystem Enabled set data request. See "40.79. Subsystem Enabled" on page 902.

➤ **To set the active sonar mode:**

1. Select the entity.
2. Choose **Set > Sensors > Active Sonar Mode**. The Active Sonar Mode dialog box opens.
3. In the System ID list, select the sonar system that you are setting. The entities provided with VR-Forces just have one active sonar system.
4. In the Mode list, select a sonar mode.
5. Click OK.

40.2. Add Jamming Target

The Add Jamming Target set data request lets you specify a target for an entity to jam. You can only specify one target at a time using this set, but you can issue it multiple times to specify multiple jamming targets.

➤ **To specify a jamming target:**

1. Select the entity that has a jamming capability.
2. Choose **Set > Jammer > Add Jamming Target**. The Add Jamming Target dialog box opens.
3. Select the entity that you want to jam.
4. Click OK.

40.3. Autonomous Actions Enabled

The Autonomous Actions Enabled set data request disables or enables intelligent decision making by simulation objects. By default, simulation objects can respond to their environment by avoiding obstacles (excluding aircraft), firing on targets, and so on. In some cases, exercise participants do not want simulation objects to take any actions other

than those specifically tasked to them by plans or human players. For example, you might want a ground vehicle assigned the Move To task to proceed in a straight line to its destination, though any obstacles.

Autonomous Actions Enabled disables (or enables) these behaviors:

- Path planning on road networks, between terrain feature obstacles, or navigation meshes.
- Dynamic collision and obstacle avoidance.
- Advanced ground movement behaviors.
- Automatic firing at other entities.
- Reacting to being fired upon. That is, there is no suppression or fleeing.

Aircraft entities do not have obstacle avoidance. The independent intelligent behavior for aircraft, when enabled, does not attempt to avoid collisions with obstacles in their flight path.

➤ **To enable or disable independent intelligent behaviors:**

1. Select the simulation object.
2. Choose **Set > Action > Autonomous Actions Enabled**. The Autonomous Actions Enabled parameters display.
3. In the Autonomous Actions Enabled list, select Yes or No.
4. Click OK.

You can also simply click the Enable/Disable Autonomous Actions button () on the Sets Toolbar or Last Selected Object Panel.

40.4. AIS Data

You can set the AIS (automatic identification system) data for surface entities. The information displays in the AIS page of the Information dialog box. All values are optional.

The values you can specify are:

- MMSI. The Maritime Mobile Service Identity is a unique nine-digit number that automatic identification systems (AIS) use to identify a vessel.
- IMO Number. The International Maritime Organization number begins with the prefix "IMO" and is followed by a unique seven-digit number. It is also used to identify a vessel.
- ETA. The estimated date and time at which the entity will arrive at its destination.
- Destination. The surface objects destination.
- Navigation Status. Select the vessel's navigation status from the list.

➤ **To set the AIS data for a surface entity:**

1. Select the entity.
2. Choose **Set > AIS data**. The AIS Data parameters display.
3. Specify the values for the entity.
4. Click OK.

40.5. Aiming

The Aiming set data request tells an entity to aim at a point, an object, or an angle. It does not cause the entity to shoot. If the entity acquires a target, it aims at the target and fires. Then it returns to the aiming behavior specified by the set data request.

An entity can move while it is aiming. If an entity is moving and it can no longer reasonably aim in the specified direction, it stops aiming. For example, if the entity is aiming at a point and the entity's movement causes the point to be behind it, the entity stops aiming (unless it has a turret that can aim in that direction). If the entity stopped aiming while it was moving, when it stops moving, it turns to aim in the specified way.

You can also use this set data request to stop the aiming behavior.

➤ **To have an entity aim at a location or object:**

1. Select the entity.
2. Choose **Set > Engagement > Aiming**. The Aiming dialog box opens.
3. Select the aiming option you want.
4. Specify the appropriate parameter for the aiming choice:
   - Aim at Point – specify the location.
   - Aim at Object – select the object.
   - Aim at Angle – specify the azimuth and elevation.
   - Cease Aiming – no parameters required.
5. Click OK.

40.6. Aircraft Lighting

The Aircraft Lighting set data request causes a fixed-wing aircraft to publish its lighting information as specified. The 3D visuals on the fixed-wing model reflect the lighting if the model is configured with those lights.

For tankers, there is another set data request. See "40.85. Tanker Lighting" on page 904.

➤ **To set the lights on a fixed-wing aircraft:**

1. Select the entity.
2. Choose **Set > Appearance > Aircraft Lighting**. The Aircraft Lighting parameters display.

3. Specify the settings for the lights.
4. Click OK.

40.7. Altitude

The Altitude request changes the altitude of simulation objects immediately. If you want a fixed-wing or rotary-wing entity to move to an altitude, assign a Move to Altitude or Fly Altitude task or use the Entity Control Panel. You can set the altitude manually or place an entity at a specific level in a multi-level terrain feature, such as a building.

➤ **To set an entity's altitude:**

1. Select the entity.
2. Choose **Set > Position > Altitude**, or click the Altitude button () on the Sets Toolbar or the Last Selected Object panel. The Altitude dialog box opens. It displays the entity's current altitude.
3. Select MSL (mean sea level) or AGL (above ground level) to specify the base for the altitude value.
4. Type the new altitude.
5. Click OK.

40.8. Appearance

You can set appearance attributes for simulation objects. The appearance attributes that are available for a simulation object are determined by its entry in *./appData/settings/vrfGui/SISO-REF-010.xml*. The default list of appearance attributes is based on the DIS standard.

The Lifeform State field of the Appearance dialog box sets the posture for an entity. If VR-Forces does not support a selected posture, it simulates the entity using the closest similar posture. If a posture involves walking or running, VR-Forces chooses the posture to use based on the entity's speed. For more information about setting posture, see "40.59. Posture (Human)" on page 892.

If you want to set the appearance of a human character, use the DI-Guy Appearance set data request, described in ("40.25. DI-Guy Characteristics (Appearance, Head, Body Weight)" on page 876).

> In aggregate-level scenarios, the Appearance set data request is included in the list of commands for the Send Radio Set task and the list of commands available on the Command menu in plans. However, it is not supported and if used, will fail. Its presence is due to the lack of context sensitivity for these menus.

➤ **To set appearance values:**

1. Select the simulation object.
2. Choose **Set > Appearance > Appearance**. The Appearance dialog box opens. The settings displayed are the current settings for the simulation object.
3. Change appearance settings as desired.
4. Click OK.

40.9. Armed

IEDs, car bombs, and suicide bombers must be armed to detonate. They are all armed by default. You can disarm them and re-arm them.

IEDs, car bombs, and suicide bombers have radios. Therefore, you can arm and disarm them using the Send Radio Set task.

➤ **To arm or disarm an explosive device:**

1. Select the explosive device.
2. Choose **Set > Engagement > Armed**. The Armed dialog box opens.
3. Select an option from the list.
4. Click OK.

40.10. Articulated Part

The Articulated Part set data request allows you to set the position of a simulation object's articulated part at runtime. This applies only to articulated parts that are not controlled by another system.

VR-Forces does not support manually articulating a part that is controlled by a system because that would cause conflicts between a set controller and the system.

➤ **To set the position of an articulated part:**

1. Select the simulation object.
2. Choose **Set > Appearance Articulated Part**. The Articulated Part dialog box opens.
3. Select the articulated part.
4. Specify the translation or rotation for the articulated part.
5. Click OK.

40.11. Assume Unknown is Hostile

The Assume Unknown is Hostile set data request means that a simulation object associated with an engagement zone can assume that any unknown simulation object that enters the engagement zone is hostile.

➤ **To set unknown objects in an engagement zone to be hostile:**

1. Select the simulation object.
2. Choose **Set > Disposition > Assume Unknown is Hostile**. The Assume Unknown is Hostile dialog box opens.
3. Select No or Yes.
4. Click OK.

40.12. Attached

Use the Attached set data request to attach a tactical graphic to an entity. You can specify the relative orientation for the graphic:

- Entity Orientation. The tactical graphic moves as the heading, pitch, and roll of the entity changes.
- World Orientation. The attached tactical graphic maintains a specific distance and altitude in the world, regardless of the entity's orientation or altitude.
- Entity Heading. The tactical graphic moves as the entity's heading and altitude changes. Its pitch and roll remain constant.

➤ **To set the tactical graphic's attachment to an entity:**

1. Select the tactical graphic.
2. Choose **Set > Disposition > Attached**. The Attached dialog box opens.
3. In the Name box, enter the entity to which you want to attach the tactical graphic.
4. Select the orientation you want.
5. Click OK.

40.12.1 Detaching a Tactical Graphic

You can also use the Attached set data request to detach a tactical graphic from an entity. This method works for both tactical graphics attached using this set data request and those attached using the methods described in "47.8. Attaching Tactical Graphics to a Simulation Object" on page 1028.

➤ **To set the tactical graphic's attachment to an entity:**

1. Select the tactical graphic.
2. Choose **Set > Disposition > Attached**. The Attached dialog box opens.
3. Select the Detach From Parent check box.
4. Click OK.

40.13. Attention Grabbing

The Attention Grabbing set data request specifies how interesting an object should be to humans as they randomly pay attention to other objects in their environment.

➤ **To set the level of attention for an object:**

1. Select the object whose level of attention you want to set.
2. Choose **Set > Disposition > Attention Grabbing**. The Attention Grabbing dialog box opens.
3. Select an option for the amount of attention that this object should receive.
4. Click OK.

40.14. Automatically Close Doors

If Automatically Close Doors is enabled, when a human entity goes through a door that can be opened or closed and there is no one else behind it, the door closes automatically. Default: enabled.

You can also close doors with the Close Door task ("32.6. Close Door" on page 651). For more information about opening and closing doors, see the chapter *Dynamic Terrain* in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To enable or disable automatically closing doors:**

1. Select the entity.
2. Choose **Set > Action > Automatically Close Doors**. The Automatically Close Doors dialog box opens.
3. Select Yes to enable, No to disable.

40.15. Base

Use the Base set data request to assign a base aerodrome to an aircraft. The assigned Base displays in the Last Selected Object Panel and the State Data page of the aircraft's Information dialog box. You can task an aircraft to return to its assigned base; see "30.41. Return To Base" on page 611.

This set data request requires use of a terrain database that has aerodrome data. For details, see "Chapter 67. Aerodromes" on page 1291.

➤ **To assign a base aerodrome to an aircraft:**

1. Select the aircraft entity.
2. Choose **Set > Base**.
3. In the Aerodrome box, enter the ICAO code of the aerodrome. You can type the code, select an aerodrome in the Objects List, or select from the list of aerodromes at the bottom of the window. In the last case, any managed aerodromes are listed at the top of the list, and then all other aerodromes in the terrain are listed below those.
4. Click OK.

ⓘ You can also assign an aircraft to a base from the Edit dialog box for that entity. As with the Base set data request, there must be an aerodrome in the terrain.

40.16. Callsign

You can specify a callsign for an entity or a unit. By default, the NETN callsign attribute uses a simulation object's marking text. The Callsign set data request allows you to define a custom callsign for a simulation object. The attribute is published on the network.

➤ **To set the NETN Callsign attribute for a simulation object:**

1. Select the entity or unit.
2. Choose **Set > Disposition > Callsign**. The Callsign parameters display.
3. In the Callsign box, enter the desired value.
4. Click OK.

40.17. Capabilities

The capabilities that are available for a simulation object are determined by its entry in *./appData/settings/vrfGui/SISO-REF-010.xml*. The default list of capabilities is based on the DIS standard.

➤ **To set simulation object capabilities:**

1. Select the simulation object.
2. Choose **Set > Disposition > Capabilities**. The Capabilities dialog box opens. It displays the current settings for the simulation object.
3. Change the settings you want to change.
4. Click OK.

40.18. Collision Avoidance Enabled

You can completely disable collision avoidance for a ground entity. You might want to disable this behavior when you need more direct control over the entity during movement planning tasks.

➤ **To disable or enable collision avoidance:**

1. Select a ground entity.
2. Choose **Set > Action > Collision Avoidance Enabled**. The Collision Avoidance Enabled parameters display.
3. From the Enabled list, choose Yes to enable collision avoidance or No to disable it.
4. Click OK.

40.19. Concealed

The Concealed request sets a simulation object's concealment attribute to True or False. For entity-level scenarios, this only affects the DIS or RPR FOM data sent over the network. It does not affect the simulation. For aggregate-level scenarios, when a simulation object is concealed, its sensor signature is changed so that it is harder to detect.

➤ **To set the concealment attribute:**

1. Select the simulation object.
2. Choose **Set > Disposition > Concealed**. The Concealed dialog box opens.
3. Select an option from the list.
4. Click OK.

40.20. Countermeasures Auto Launch

The Countermeasures Auto Launch set data request lets you enable and disable automatic launching of countermeasures. If you disable automatic countermeasures, you can still launch them manually using the Launch Countermeasures task ("31.25. Launch Countermeasures" on page 640).

➤ **To enable or disable automatic launch of countermeasures:**

1. Select the entity.
2. Choose **Set > Engagement > Countermeasures Auto Launch**. The Auto Launch dialog box opens.
3. Select an option from the list.
4. Click OK.

40.21. Current Speed

The Current Speed set data request sets the speed of a simulation object immediately and changes the ordered speed to this new speed value. By contrast, Ordered Speed ("40.57. Ordered Speed" on page 891) causes a simulation object to accelerate or decelerate to the desired speed.

A negative value causes the entity to go backwards if it supports backwards movement.

For all simulation objects except humans and fixed-wing aircraft that are in the air, if you set the current speed and the object does not have a task, it begins to move at that speed and then slows down until the speed reaches 0. Fixed-wing aircraft that are in a default orbit mode change their speed to the new current speed. Humans might take a step, but then will stop moving.

➤ **To set current speed:**

1. Select the simulation object.
2. Choose **Set > Action > Current Speed**. The Current Speed dialog box opens.
3. In the Current Speed text box, set the speed.
4. Click OK.

40.22. Damage Percentage

Use the Damage Percentage set data request to specify that a surface watercraft capable of sustaining multiple hits has sustained a certain amount of damage. For details, see "26.2. Multi-hit Actuator" on page 522.

➤ **To set an object's percent damage:**

1. Select the simulation object.
2. Choose **Set > Disposition > Damage Percentage**. The Set Damage Percentage parameters display.
3. Enter the percent damage and then click OK.

40.23. Destroyed

You can set a simulation object's state to be destroyed. This can be helpful when you want a simulation object to be destroyed without having to first simulate its destruction by another simulation object.

➤ **To set an object's state to destroyed:**

1. Select the simulation object.
2. Choose **Set > Disposition > Destroyed**. The simulation object is immediately destroyed.

40.24. Detonation Fuse Type

The Detonation Fuse Type set data request lets you configure detonation of improvised explosive devices (roadside IED entity, drone with munition), suicide bombers, and car bombs. You can set them to detonate immediately, at a certain time delay, or based on the proximity of a target entity. If you specify proximity detonation, you can set the radius of the proximate area. If you specify time, you must set the time in seconds until detonation. Detonation Fuse Type implicitly arms the device. (For details about arming devices, see "40.9. Armed" on page 870.)

IEDs, car bombs, and suicide bombers have radios. Therefore, you can detonate them using the Send Radio Set task.

➤ **To set the detonation fuse type:**

1. Select the object that you want to configure.
2. Choose **Set > Engagement > Detonation Fuse Type**. The Detonation Fuse Type dialog box opens.
3. In the Fuse Type list, select the fuse type (Timed, Immediate, or Proximity).
4. If you selected a Timed fuse, in the Time box, specify the elapsed time, in seconds, at which the IED should detonate.

5. If you selected a Proximity fuse, in the Proximity box, specify the radius of the area in which a simulation object will trigger the detonation.
6. Click OK.

40.25. DI-Guy Characteristics (Appearance, Head, Body Weight)

The DI-Guy Characteristics set data request determines the clothing and physical characteristics of a lifeform when it is displayed in a 3D simulation that uses DI-Guy characters.

- The DI-Guy Characteristics set data request applies only to lifeform entities.
- The Apply Selected Appearance Immediately check box gives you the option to update the selected entity with the new appearance as you make selections. When the check box is selected, clicking Cancel does not reset the appearance to what it was before you made changes. You can only see the changes in the 3D view.

➤ **To set DI-Guy appearance, head, or body weight:**

1. Select the entity.
2. Choose **Set > Appearance > DI-Guy Characteristics**. The DI-Guy Characteristics dialog box opens.
3. Optionally, select the Apply Selected Appearance Immediately check box. This makes the changes before you click OK.
4. Optionally, filter the appearance list by choosing options in the various filter lists.
5. To change the appearance, select an appearance from the DI-Guy Appearance list.
6. To change the head, select an option in the DI-Guy Head list.
7. To change the weight of the character, slide the DI-Guy Body Weight slider or change the value in the text box (the range is 0.5 - 5.0).
8. Click OK.

40.26. DI-Guy Hand Item

The DI-Guy Hand Item set data request lets you change the hand item that a character has. DI-Guys can have a variety of hand items. For example, a civilian can have a cell phone, a pistol, or a camcorder, while a soldier can have a variety of weapons.

➤ **To set a DI-Guy's hand item:**

1. Select the entity.
2. Choose **Set > Appearance > Hand Item**. The DI-Guy Hand Item dialog box opens.
3. Select a hand item from the list.

4. Optionally, in the 3D view only, select the Apply Selected Hand Item Immediately check box. This displays the selected entity with the new hand item so that you can see what it looks like before you apply the change.
5. Click OK.

40.27. Disembarked

The Disembarked set data request immediately disembarks a simulation object (as opposed to simulating disembarkation, as done by the Disembark task). It is the equivalent of the **Objects > Disembark** command. However, Disembarked allows you to include immediate disembarkation in a plan.

➤ **To set a simulation object to be disembarked:**

1. Select the simulation object. (If you are using the 2D icon model, embarked simulation objects are not displayed, so you have to select it in the Objects List Panel.)
2. Choose **Set > Embarkation > Disembarked**.

40.28. Embarked

The Embarked set data request immediately embarks a simulation object (as opposed to simulating embarkation, as done by the Embark task). It is the equivalent of the **Objects > Embarkation > Embark On** command. However, Embarked allows you to include immediate embarkation in a plan.

The default location for this set data request is the center of the parent simulation object (0,0,0). It does not use preconfigured slots like the Embark task does. If you are using a Stealth observer mode and want placement of embarked simulation objects to look good, you will have to calculate offsets for each simulation object you embark.

⚠ If you are using HLA and want to use the embarkation feature, you must use RPR FOM 2, draft 17 or later.

➤ **To set a simulation object to be embarked:**

1. Select the simulation object to be embarked.
2. Choose **Set > Embarkation > Embarked**. The Embarked dialog box opens.
3. Select the simulation object to embark on.
4. Optionally, specify an embarkation offset.
5. Click OK.

40.29. Emergency Beacon

The US Air Force human entity is equipped with an emergency transponder, which can be turned on or off by using the Emergency Beacon set data request. When the transponder is on, it is easier for an entity with the Search and Rescue Sensor to identify. Even if the transponder is off, however, the sensor can detect it when the sensor is close to the transponder. (Detecting the transponder is limited by the sensor's maximum range.)

➤ **To turn an emergency beacon on or off:**

1. Select the entity.
2. Choose **Set > Sensors > Emergency Beacon**. The Emergency Beacon dialog box opens.
3. From the Emergency Beacon list, select either On or Off.
4. Click OK.

40.30. Emitter

The Emitter set data request enables or disables the publication of emitter data for simulation objects that have emitters. This models basic electromagnetic emissions.

Some sensors have an active emitter system that you must enable for the sensor to detect targets. The emitters publish one or more beams with the sensor-specific data to the network. That information displays in the entity's information dialog under Emitters. Currently, three types of sensors use emitter systems: active radar, jamming radar, and active sonar.

Emission properties (azimuth, elevation, effective radiated power, and so on) for each beam are specified in the sensor component descriptor in the object parameter database. Emission properties are published by the sensor component. A sensor component can publish its emission data as a standard beam type or a tracking beam type.

- Setting a target does not cause the beam to actually track the target. The beam's behavior is determined by the parameters in its description in the object parameter database.
- The emitter continues to publish the targeted entity as its tracking target until you set another target, regardless of the status of that entity in the exercise. There is no way to set the target to "none".
- See "80.1. Configuring Emitters" on page 1572 for an explanation of how to configure individual and multiple emitters in the object parameter database.

➤ **To turn on electromagnetic emissions:**

1. Select the simulation object.
2. Choose **Set > Sensors > Emitter**. The Emitter dialog box opens.
3. Select the Radar ID from the list, or you can choose All.
4. In the Emitter list, select On.
5. Click OK.

40.31. Engagement Zones

The Engagement Zone set data request lets you associate one or more engagement zones with a simulation object.

➤ **To associate a simulation object with an engagement zone:**

1. Select the simulation object.
2. Choose **Set > Engagement > Engagement Zones**. The Engagement Zones dialog box opens.
3. Select one or more engagement zones in the list.
4. Click OK.

40.32. Force

You can change a simulation object's force during runtime. This might be useful in a simulation in which neutral forces begin assisting one force or another, or in which opposing forces capture friendly force weaponry and begin using it. You can also set the force for combat engineering objects in aggregate-level scenarios.

When you change a simulation object's force, its:

- Echelon ID changes to reflect the new force.
- Icon changes to the appropriate force color and type.
- Force ID changes.

The entity enumeration does not change.

➤ **To set the force of an object:**

1. Choose **Set > Disposition > Force**. The Force dialog box opens.
2. Select a force from the list.
3. Click OK.

40.33. Formation

VR-Forces supports these formations:

- Column (default)
- Line
- Wedge
- Vee
- User-defined formations

Formations are defined in formation files in the directory *./data/simulationModelSets/<model_set>/vrfSim/formation*. Formation files have the extension *.frm*. In addition to the named formations provided by MAK, VR-Forces supports user-defined formations. If you create a new formation in the Simulation Object Editor, it

is automatically added to the formation list for the unit types that support it. For information about configuring formations and about how to create user-defined formations, see "79.1. Configuring Formations" on page 1566.

> The vee and wedge formations provided with VR-Forces place the unit leader at the end of one of the arms of the formation. If a unit has fewer than seven or eight members, it will not be symmetrical.

When you set the formation for a unit, the simulation objects that make up the unit snap to formation immediately. If you want simulation objects to simulate moving into formation, rather than snapping to formation, use one of the formation movement tasks. For more information, see "30.26. Move Into Formation" on page 603 and "30.52. Transition Into Formation" on page 621.

➤ **To set a unit's formation:**

1. Select the unit whose formation you want to set.
2. Choose **Set > Unit > Formation**. The Formation dialog box opens.
3. In the Formation list, select the formation that you want the unit to use.
4. Click OK.

40.34. Freeze Fuel Levels

You can set an entity to stop depleting fuel by freezing the resource. The entity behaves normally in the scenario; it simply no longer expends its fuel resources as it moves.

➤ **To freeze or unfreeze an entity's fuel levels:**

1. Select the entity.
2. Choose **Set > Action > Freeze Fuel Levels.** The Freeze Fuel Levels dialog box opens.
3. In the Freeze Fuel Levels list, select Yes or No.
4. Click OK.

40.35. Freeze Movement

The Freeze Movement request freezes an entity's movement system. It prevents any physics- or simulation-related change to the object's spatial attribute (location in the simulated world). The end user can move the entity within the UI, but during the normal simulation the entity publishes the same location and orientation.

Tasks that require an entity to move, such as Move to Location, pause while Freeze Movement is set to Yes. The entity's other systems, such as weapons, remain active, and the entity will attempt reactive tasks that do not require a spatial change. For example, a tank's turret may move to fire, a soldier may move his arms to aim a weapon, and so on.

➤ **To freeze or unfreeze an entity's movement:**

1. Select the entity.
2. Choose **Set > Action > Freeze Movement**. The Freeze Movement dialog box opens.

3. In the Freeze Movement list, select Yes or No.
4. Click OK.

40.36. Freeze Munition Levels

You can set an entity to stop depleting munitions by freezing the resource. The entity behaves normally in the scenario; it simply no longer expends munitions while using its weapon(s). This includes countermeasures.

➤ **To freeze or unfreeze an entity's munition levels:**

1. Select the entity.
2. Choose **Set > Action > Freeze Munition Levels**. The Freeze Munition Levels dialog box opens.
3. In the Freeze Munition Levels list, select Yes or No.
4. Click OK.

40.37. GPS Jammer

The GPS Jammer set data request enables and disables GPS jamming for an entity equipped with the GPS Jammer system. This causes air entities with the GPS Receiver system—such as aircraft, UAVs, and missiles—to perceive that their current location is different from that of their ground truth and, as a result, go off course.

The offset vector that determines how the jammed entity's perceived location differs from its true location is calculated based on:

- The power of the GPS jammer
- The distance of the jammed entity from the jamming entity

➤ **To set the jammer state:**

1. Select the entity.
2. Choose **Set > Jammer > GPS Jammer**. The Set GPS Jammer parameters display.
3. From the Operational Status list, select On or Off.
4. Click OK.

40.38. Ground Clamp Fidelity

By default, individual human entities clamp to the ground with high fidelity, so that they move around on the terrain in a realistic manner. You can set human entities to clamp with low fidelity, clamping them to the navigation mesh instead of the terrain. This improves performance but the human characters can appear to move through the terrain instead of on it. For performance reasons, when you add a crowd simulation object, the entities in it default to low fidelity clamping.

Low fidelity clamping only works in navigation areas. If there is no navigation area specified for the terrain, the human entities use high fidelity clamping.

➤ **To set the ground clamp fidelity for human entities:**

1. Select the entity or entities.
2. Choose **Set > Action > Ground Clamp Fidelity**. The Ground Clamp Fidelity dialog box opens.
3. In the Ground Clamp Fidelity list, select Low or High.
4. Click OK.

40.39. Heading

The Heading request instantly sets a simulation object's heading. If you want a simulation object to move to a new heading by pivoting or executing a K-turn, use the Turn To Heading task. You can also set a simulation object's heading manually. For details, see "14.5. Specifying an Object's Heading Dynamically" on page 389.

➤ **To set an object's heading:**

1. Select the simulation object.
2. Choose **Set > Position > Heading**, or click the Heading button () on the Sets Toolbar or the Last Selected Object panel. The Heading dialog box opens. It displays the simulation object's current heading.
3. In the Heading text box, enter a heading.
4. In the Relative To list, select one of:
   - True North. Turn to the specified absolute heading.
   - Magnetic. Turn to the specified heading relative to magnetic north.
   - Host Object (if embarked). If embarked, turn to the specified heading relative to the host simulation object.
   - Current Heading. Turn the specified number of degrees relative to the current heading.
5. Click OK.

40.40. Hide Occupants

The Hide Occupants set data request overrides the Show Occupants setting on the Entity Display Settings page for the entity for which you set it. VR-Forces can automatically a human in the driver or pilot's seat of a vehicle. These occupants are not simulated objects; they are simply visualized. You might want to disable this setting for performance reasons or because you are embarking a simulated person in the vehicle. You can also set a vehicle to display the occupant that you previously set to be hidden.

For more information about the global setting for displaying occupants in a vehicle, see "51.4. Displaying DI-Guys in Vehicles" on page 1085.

➤ **To display or hide occupants of a vehicle:**

1. Select the simulation object.
2. Choose **Set > Appearance > Hide Occupants**. The Hide Occupants dialog box opens.
3. Select an option in the Occupant Visibility list.
   - Hide Occupants. Hides the driver or pilot.
   - Show Occupants. Displays the occupant that was previously set to hidden.
4. Click OK.

40.41. IFF

Aircraft can model a NATO Identification Friend or Foe (IFF) transponder.

- VR-Forces implements the NETN FOM, based on RPR FOM 2.0, with IFF extensions from RPR FOM 3.0. If you use HLA Evolved, use the FOM provided by MAK.
- The dialog box does not force you to enter valid codes. It truncates invalid entries to the correct length (but not necessarily valid data).
- When you open the dialog box, it reflects the current IFF settings for the entity, if any.
- You can view an entity's IFF settings on the IFF tab of its Information dialog box.

If an aircraft has an IFF Code Changer system and it enables the IFF Code Changer reactive task, its IFF code changes every 30 minutes based on the entries in a CSV file. By default, entities are not configured with this system and the task is inactive. A sample CSV file is provided: *./data/simulationModelSets/EntityLevel/extra/iff-codes.csv*.

➤ **To set the IFF transponder:**

1. Select the entity.
2. Choose **Set > Sensors > IFF**. The appropriate IFF dialog box opens (either Figure 295 or Figure 296 on the next page).

Figure 295. IFF dialog box (military aircraft)

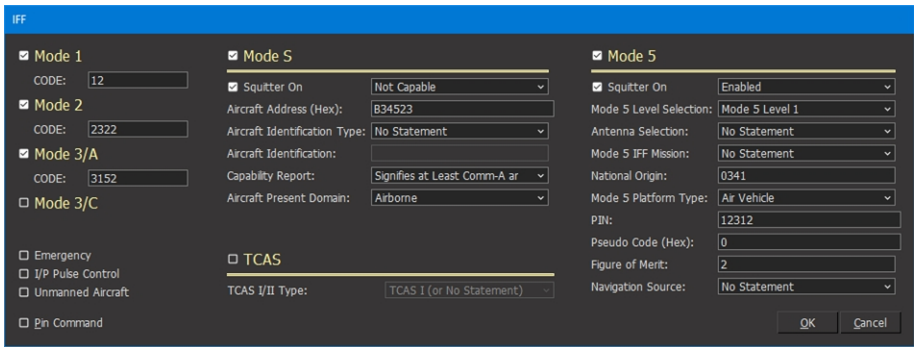
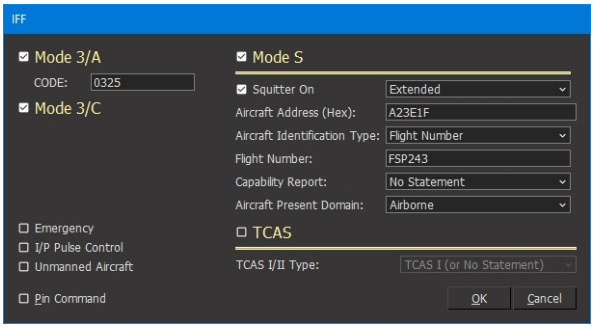


Figure 296. IFF dialog box (civilian aircraft)



3. Fill out fields as desired.
4. Click OK.

You can use this set data request for an entity that does not exist yet, such as an entity that will be created by a Spawn Pattern's spawn point. In that case, the request assumes that all IFF modes are supported by the entity, even if the created entity does not.

40.42. Invisible

The Invisible set data request causes simulation objects to be invisible on the map display in 2D or 3D. They are still listed in the Objects List Panel.

➤ **To set a object to be invisible:**

1. Select the simulation object.
2. Choose **Set > Appearance > Invisible**. The Invisible dialog box opens.
3. Select an option from the Invisible list.
4. Click OK.

40.43. Invulnerable

If a simulation object is invulnerable, it does not take damage from munitions.

➤ **To set an object to be invulnerable:**

1. Select the entity.
2. Choose **Set > Disposition > Invulnerable**, or click the Invulnerable button ( ). The Invulnerable dialog box opens.
3. Select an option from the list.

40.44. Jammer State

The Jammer State set data request enables and disables radar jamming for an entity. For more information about jamming, see "9.6. Electronic Warfare" on page 306.

➤ **To set the jammer state:**

1. Select the entity.
2. Choose **Set > Jammer > Jammer State**. The Jammer State parameters display.
3. From the State list, select On or Off.
4. Click OK.

40.45. Jamming Mode

The Jamming Mode set data request lets you specify the emitter ID for a jammer and the mode it uses. The options are:

- Deception jamming
- Mixed mode jamming
- Mixed noise

For more information about jamming, see "9.6. Electronic Warfare" on page 306.

➤ **To specify the jamming mode:**

1. Select the jamming entity.
2. Choose **Set > Jammer > Jamming Mode**. The Jamming Mode dialog box opens.
3. The Emitter ID is Jamming Emitter.
4. Select a mode in the Mode list.
5. Click OK.

40.46. Laser Code

The Laser Code set data request sets the code for laser beams sent by an entity and the code that the entity's missiles seek when they fire. By default, VR-Forces calculates a laser code for each laser-capable entity when it is created. You can change the laser code to a code of your choice. For information about the consequences of setting laser codes, see "9.4.2 Lasing Targets" on page 304.

➤ **To set a laser code:**

1. Select the entity.
2. Choose **Set > Laser Designator > Laser Code**. The Laser Code dialog box opens.
3. Type a number between 1111 and 1788, using only the digits 1-8.
4. Click OK.

40.47. Laser Power and Wavelength

The Laser Power and Wavelength set data request lets you specify the power and wavelength for a laser designator. VR-Forces does not use these settings when it simulates laser designators. However the information is sent over the network for use by applications that can use it. For example, the color of the laser designator can be set based on the wavelength if it is visualized in another application.

The dialog box does not report the current value, so if you only want to change one value, set the other one to -1 to keep the current value.

➤ **To set the laser power and wavelength for a lasing entity:**

1. Select the entity.
2. Choose **Set > Laser Designator > Laser Power and Wavelength**. The Laser Power and Wavelength dialog box opens.
3. Specify the Power value, in watts. Setting it to -1 preserves the current value.
4. Specify the Wavelength value, in microns. Setting it to -1 preserves the current value.
5. Click OK.

40.48. Loadout

The F/A-18 Hornet is able to use the Loadout set data request, which changes the number of different kinds of missiles currently loaded on the entity. This enables you to change the missiles available for use by the entity during the course of a simulation.

You can use *./data/simulationModelSets/EntityLevel/scripts/set-loadout.lua* as an example for creating your own script to change the current missile loadouts for an entity.

Table 41. Loadout settings

Loadout	Missile Distribution
Default	• Maverick = 2 • Sidewinder = 2 • AMRAAM = 2 • Harpoon = 0
Intercept	• Maverick = 0 • Sidewinder = 2 • AMRAAM = 6 • Harpoon = 0

Table 41. Loadout settings (continued)

Loadout	Missile Distribution
Surface	• Maverick = 0 • Sidewinder = 0 • AMRAAM = 2 • Harpoon = 4
Ground	• Maverick = 4 • Sidewinder = 0 • AMRAAM = 2 • Harpoon = 0

➤ **To set the missile loadout for an F/A-18 Hornet:**

1. Select the entity.
2. Choose **Set > Disposition > Loadout**. The Loadout dialog box opens.
3. Specify the Loadout. Table 41 shows the missile counts in each setting.
4. Click OK.

40.49. Location

To move a simulation object instantly from one place to another, you can set its location or you can drag it to a new location. For information about dragging simulation objects, see "14.9.1 Dragging an Object to a New Location" on page 392. For ground entities, specifying an altitude lets you place an entity at different levels (floors) in buildings that have interior geometry.

➤ **To set location:**

1. Select the simulation object.
2. Choose **Set > Position > Location**. The Location dialog box opens. It displays the simulation object's current location. The cursor changes to input mode.
3. Specify the location where you want to move the simulation object. You can click on the terrain or expand the location group box by clicking the arrow next to the coordinates and altitude to enter a specific location.
4. Optionally, set the altitude, as described in "14.4. Specifying an Object's Altitude" on page 386.
5. Click OK.

40.50. Look At

The Look At set data request tells a human entity to look at a point, an object, or an angle. This set causes the entity's head to move, not its body. It will move up to 90 degrees to either side (azimuth) and up to 45 degrees of elevation (where looking straight ahead is 0 degreess). If the target object or location is at a greater angle than the maximum allowed amount, the head does not turn.

> The range of head motion is configurable in *./data/simulationModelSets /EntityLevel/vrfSim/platforms/human.ope* in the **look-at-actuator**. It is specified in radians.

You can also use this set data request to stop the look at behavior.

➤ **To have an entity look at a location or object:**

1. Select the entity.
2. Choose **Set > Action > Look At**. The Look At dialog box opens.
3. Select the Look At option you want.
4. Specify the appropriate parameter for the aiming choice:
   - Look at Point – specify the location.
   - Look at Object – select the object.
   - Look at Angle – specify the azimuth (heading) and elevation.
   - Cease Looking – no parameters required.
5. Click OK.

40.51. Markings

Some vehicles are configured with texture maps that let you dynamically change exterior markings, such as a national flag, unit type, and other emblems. You can change the markings at run time. Figure 297 shows a C-130 aircraft with U.S.A.F. markings and desert camouflage, and the same model with R.A.F. markings and forest camouflage.

Figure 297. Changing markings



If you assign this set to a unit, the markings are passed to its subordinates.

For details about configuring markings, please see "69.2.9 Specifying Visual Markings" on page 1357, and also "Adding Dynamically Changeable Markings to Models" in the *Adding Content to MAK ONE Applications Reference Manual*.

➤ **To set a simulation object's markings:**

1. Select the entity.
2. Choose **Set > Appearance > Markings**. The Markings dialog box opens.
3. In each list, select the markings that you want to set for this object.
4. Click OK.

40.52. Name Tag

For soldier entities, you can set a name tag to display on the soldier's chest. The default is "SMITH". The name tag can display up to 31 characters. It is case sensitive, displaying both upper- and lower-case characters. At this time, you cannot specify a name for all soldier characters.

The Update Name Tag Immediately check box allows you to specify whether you want the name to update as you enter it. If this is selected, you might notice some flickering while you type. If you clear the check box, the name displays when you click OK.

You can also specify a rank and unit, as described in "40.63. Rank Patch" on page 894 and "40.94. Unit Patches" on page 908.

➤ **To specify a soldier's name:**

1. Select the soldier entity.
2. Choose **Set > Appearance > Name Tag**. The Name Tag parameters display.
3. Optionally, clear the Update Name Tag Immediately check box.
4. Enter the desired name.
5. Click OK.

40.53. Navigation Enabled

By default, humans use the navigation data in navigation areas to plan movement. Use the Navigation Enabled set data request to disable use of navigation data.

Navigation is disabled on a per-entity basis. If you disable navigation for an entity, other entities continue to treat that entity as an obstacle for navigation purposes.

➤ **To enable or disable use of navigation data:**

1. Select the entity.
2. Choose **Set > Action > Navigation Enabled**. The Navigation Enabled dialog box opens.
3. Select an option from the Enabled list.
4. Click OK.

40.54. Navigation Preferences

The Navigation Preferences set data request allows you to specify that when an entity plans its path using navigation data, it should take into account road data when it is available.

When an entity is set to Prefer Roads, the entity gives preference to all segments along the preferred graph over those on the non-preferred movement graph. This is typically used to have vehicles prefer movement along roads, rather than cutting across open terrain, even if the distance along roads is longer. Civilian vehicles typically default to preferring roads.

If the movement mode is Avoid Roads, entities do not attempt to use roads. They plan a path that is as direct as possible while avoiding obstacles. Military vehicles typically default to ignoring roads.

You can configure an entity to start with a navigation preference or to ignore roads by editing the **default-preference** parameter in the **navigation-preference** controller in the movement system definition file for this entity type.

➤ **To set navigation preferences:**

1. Select the entity.
2. Choose **Set > Action > Navigation Preferences**. The Navigation Preferences dialog box opens.
3. To have the entity take road data into account, select Prefer Roads or Ignore Roads from the Navigation Preferences list. To have an entity plan its path as if there were no road data, select Ignore Roads.
4. Click OK.

40.55. Notify Level

You can set the notification level for simulation object console messages without opening the Information dialog box. You can also set the notification level for combat engineering objects in aggregate-level scenarios.

➤ **To set the notification level:**

1. Choose **Set > Other > Notify Level**. The Notify Level dialog box opens.
2. Select a notification level in the list.
3. Click OK.

40.56. Orbit

The Orbit set data request lets you specify the orbit for a satellite. You can specify a Kepler orbit or a geostationary orbit. Kepler orbits are specified by two-line elements. They are available at http://celestrak.org and https://www.space-track.org.

➤ **To set the orbit for a satellite:**

1. Select the satellite.
2. Choose **Set > Position > Orbit**. The Orbit dialog box opens.
3. To set a Kepler orbit:
    a. Select the Kepler Orbit option. The input window is enabled.
    b. Paste in the two-line element that you want to use.
4. To set a geostationary orbit:
    a. Select the Geostationary Orbit option.
    b. Type the longitude of the orbit, in degrees.
5. Click OK.

40.57. Ordered Speed

The ordered speed is the speed at which you want a simulation object to move. It may not move at that actual speed due to soil type, slope, and so on. In aggregate-level scenarios, speed may be affected by modifiers. If you set the speed while a simulation object is moving, it accelerates or decelerates from the current speed to the new ordered speed, if it can.

A negative value causes the entity to go backwards if it supports backwards movement.

- Regardless of the speed that you enter, a simulation object's speed cannot exceed the value set by the **max-speed** parameter in the object parameter database.
- The Ordered Speed request sets the ordered speed of a fixed-wing entity in its current context – air or ground. The entity will not exceed the maximum speed for that context. However, if the maximum allowable ground speed is greater than the lift speed of the entity, and you assign an ordered speed greater than the lift speed, when the entity reaches that speed, it tries to take off.
- If you want a simulation object's speed to change instantly, use the Current Speed set data request ("40.21. Current Speed" on page 874).
- If a simulation object is not moving, this set data request has no effect.
- You can quickly change the ordered speed for an aircraft or marine entity using the Entity Control Panel. For details, see "29.2. The Entity Control Panel" on page 567
- During a session, the speed measurement for aircraft entities defaults to the one you last specified in the Ordered Speed dialog box. This allows you to easily use the same speed for multiple entities.

➤ **To set the ordered speed of an object:**

1. Select the simulation object.
2. Choose **Set > Action > Ordered Speed**. The Ordered Speed dialog box opens.

3. In the Ordered Speed text box, set the speed.
4. For aircraft entities, specify the measurement of the speed: Ground Speed, TAS (true airspeed), IAS (indicated air speed), or Mach.
5. Click OK.

40.58. Pitch

Some simulation objects allow you to set the pitch. The Pitch request instantly sets a simulation object's pitch. You can also set a simulation object's pitch manually. For more information, see "14.6. Specifying an Object's Pitch Dynamically" on page 389.

➤ **To set an object's pitch:**

1. Select the simulation object.
2. Choose **Set > Position > Pitch**. The Pitch dialog box opens, displaying the simulation object's current pitch.
3. In the Pitch text box, type a pitch, in degrees. You can enter a number between 90 and -90.
4. Click OK.

40.59. Posture (Human)

A human entity (civilian or dismounted infantry) has a posture that determines its position when it is not moving and its type of movement when it is moving. Table 42 lists the position and movement relationships for each posture.

Table 42. Posture relationships

Posture	When it is not moving, it is:	When it is moving, it is:
standing	standing	walking or running
kneeling	kneeling	crawling
prone	prone	crawling
running	standing	walking or running
walking	standing	walking or running
crawling	prone	crawling
parachuting	standing	walking or running
jumping	standing	walking or running
sitting	standing	walking
squatting	prone	crawling
crouching	prone	crawling
wading	standing	walking or running

- For those postures listed as "walking or running", VR-Forces sets the movement posture based on the entity's speed.
- If an entity does not support a particular posture, it is set to the closest supported posture.
- Setting posture in the Lifeform State field of the Appearance dialog box has the same effect as setting the posture using the Posture set data request.
- Some embarkation slots specify the posture of occupants.

➤ **To set a lifeform's posture:**

1. Select the lifeform.
2. Choose **Set > Action > Posture**. The Posture dialog box opens.
3. Select a posture from the list.
4. Click OK.

40.60. Preferred Trajectory Height

The Preferred Trajectory Height set data request specifies the preferred height for indirect fire munitions, such as Fire for Effect tasks and munitions that are not triggered by a task. Indirect fire tasks override the preferred trajectory height if they need to use a different trajectory to hit the target.

➤ **To specify a preferred trajectory height:**

1. Select an entity that has indirect fire capability.
2. Choose **Set > Engagement > Preferred Trajectory Height**. The Preferred Trajectory Height dialog box opens.
3. Specify the desired altitude.
4. Click OK.

40.61. Promote From Clutter

You can take a clutter object and convert it an entity, which you can then assign tasks, set data requests, and so on. Clutter objects are created by Aircraft Clutter Area and Ship Clutter Area. Once you promote it, you cannot demote it to its previous existence as a clutter object.

➤ **To promote a clutter object to an entity:**

1. Select the clutter object.
2. Choose **Set > Promote From Clutter**.

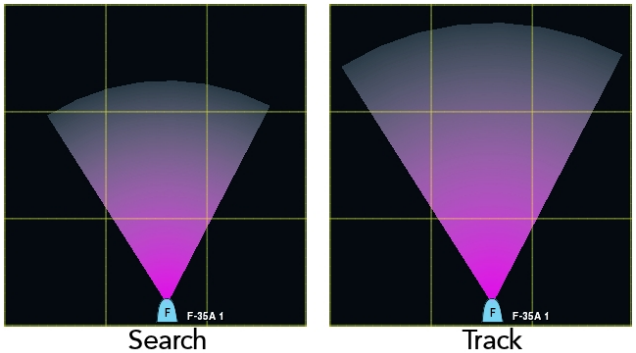
You can also click Promote from Clutter ( ) on either the Last Selected Object or Command Panel.

40.62. Radar Mode

Use the Radar Mode set data request to change the type of beam that an entity emitter publishes. By default, each fixed-wing aircraft has one emitter that has two beam types, search and track.

Figure 298 illustrates the difference in appearance between the beams for the two radar modes in the 2D view. (The beams do not display unless you enable Show Emitter Volumes in Entity Display Settings.)

Figure 298. Radar mode beams



For more information about emitter beams, see "80.1. Configuring Emitters" on page 1572.

➤ **To set the radar mode:**

1. Select the entity whose radar mode you want to set.
2. Choose **Set > Sensors > Radar Mode**. The Radar Mode dialog box opens.
3. If the entity has more than one emitter, select the emitter ID you want to set in the Emitter ID list.
4. Select the mode in the Mode list.

You can use this set data request for an entity that does not exist yet, such as an entity that will be created by a Spawn Pattern's spawn point. In that case, the Emitter ID and Mode options are free text boxes. You are responsible for entering values that are supported by the entity, otherwise the set data request will fail.

40.63. Rank Patch

For soldier entities, you can set a rank patch to display. At this time, not all soldiers are configured with ranks.

The Apply Selected Patch Immediately check box allows you to specify whether you want the patch to update when you select it. This is selected by default. If you clear the check box, the rank patch changes after you click OK.

You can also specify a name and unit, as described in "40.52. Name Tag" on page 889 and "40.94. Unit Patches" on page 908.

➤ **To specify a soldier's rank:**

1. Select the soldier entity.
2. Choose **Set > Appearance > Rank Patch**. The Rank Patch parameters display.
3. Optionally, clear the Apply Selected Patch Immediately check box.
4. Select the desired rank from the list.
5. Click OK.

40.64. Remote Sensor

The Remote Sensor set data request lets an entity use another entity's sensors to discover sensor contacts. This set data request is only available to entities that have a Remote Sensor Receiver system.

An example of using a remote sensor is the Patriot Missile Battery. The Patriot Engagement Control System has a Remote Sensor Receiver system. It receives spot reports from the Patriot Radar AN/MPQ-65.

➤ **To specify a remote sensor for an entity:**

1. Select the entity that needs to receive remote spot reports.
2. Choose **Set > Sensors > Remote Sensor**. The Remote Sensor dialog box opens.
3. Select the remote entity (the entity from which this entity wants to receive spot reports).
4. Click OK.

40.65. Remove Jamming Target

The Remove Jamming Target set data request tells an entity to stop jamming the specified target. You can only remove one target at a time using this set, but you can issue it multiple times to remove multiple jamming targets. The Remove Jamming Target dialog box lists all entities. There is no list available that only lists jamming targets.

➤ **To remove a jamming target:**

1. Select the entity that is jamming.
2. Choose **Set > Jammer > Remove Jamming Target**. The Remove Jamming Target dialog box opens.
3. Select the target to stop jamming.
4. Click OK.

40.66. Reorganize

If automatic reorganization is not enabled, you can reorganize a unit by issuing the Reorganize set data request.

➤ **To reorganize a unit:**

1. Select the unit that you want to reorganize.
2. Choose **Set > Unit > Reorganize**.

The change takes place immediately. For more information about reorganization, see "25.1.1 Reorganizing Units" on page 516.

40.67. Resources

You can specify the value to assign to a resource. In entity-level scenarios, the resources provided with VR-Forces that you can set are fuel and ammunition. The object parameter database lists the ammunition types available for each type of simulation object. In aggregate-level scenarios a simulation object might not have any resources, depending on how it is configured in the Simulation Object Editor.

➤ **To specify a resource:**

1. Select the simulation object.
2. Choose **Set > Disposition > Resources**. The Resources dialog box opens.
3. Select a resource from the Resource list. The dialog box displays the maximum amount of the resource allowed, as configured in the object parameter database.
4. Specify a value for the resource.
5. Click OK.

40.68. Restore

Restoring a simulation object returns its resources to the default values specified in the Simulation Model Set. If the simulation object is destroyed, it returns it to life.

➤ **To restore an object:**

1. Select the simulation object.
2. Choose **Set > Disposition > Restore**.

40.69. Rules of Engagement

Rules of engagement specify the conditions under which a simulation object will fire at the enemy. The options are:

- Fire at Will. If an entity detects a target, it can fire automatically.
- Fire When Fired Upon. Do not fire at detected targets unless the simulation object has already been under fire.
- Fire When In Engagement Zone. If a detected target is in an engagement zone that is associated with the firing simulation object, it may fire at the target at will.

- Fire on Priority Targets Only. Fire only at targets in the simulation object's target priority list. See "40.87. Target Priority List - Add" on page 905 to learn how to add targets to the priority list.
- Hold Fire. Do not fire at targets.

The rules of engagement for a fixed-wing entity are in effect whether it is on the ground or in the air. If you do not want fixed-wing entities to fire missiles while they are on the ground, you must change the rules of engagement.

➤ **To specify the rules of engagement:**

1. Select the simulation object.
2. Choose **Set > Engagement > Rules of Engagement**, or click the Rules of Engagement button () on the Sets Toolbar. The Rules of Engagement dialog box opens.
3. In the Rules of Engagement list, select the rule you want to apply.
4. Click OK.

40.69.1 How Fire-When-Fired-Upon Works

A simulation object considers itself under fire when:

- An "Entity Impact" detonation interaction that targeted this entity has recently (within **underFireTime** seconds (default 120)) been received.
- Any detonation has happened within **underFireDistance** (default 1000 meters) of this entity.

There is no attempt to return fire specifically at the attacking entity. If a simulation object thinks that it is under fire and is equipped with a weapon, it fires at any enemies that it can target.

The **underFireTime** and **underFireDistance** parameters are part of the under-fire-determination-sensor component in a simulation object's platform (OPE) file.

40.70. Sector Blanking

When Sector Blanking is enabled, targets within the specified sector are not spotted. This set data request is available to simulation objects that have a radar sensor. Sector blanking parameters are stored on a per-entity basis and are saved in the scenario and checkpoint files.

Sectors are defined by a start azimuth and end azimuth. They are relative to the entity orientation, with 0 degrees being directly ahead.

➤ **To enable or disable sector blanking:**

1. Select a simulation object that has a radar sensor.
2. Choose **Set > Sensors > Sector Blanking**. The Sector Blanking dialog box opens.

3. Select or clear the Enable Sector Blanking check box.
4. If you enable sector blanking, specify a start azimuth and end azimuth, with 0 degrees being directly ahead of the simulation object.
5. Click OK.

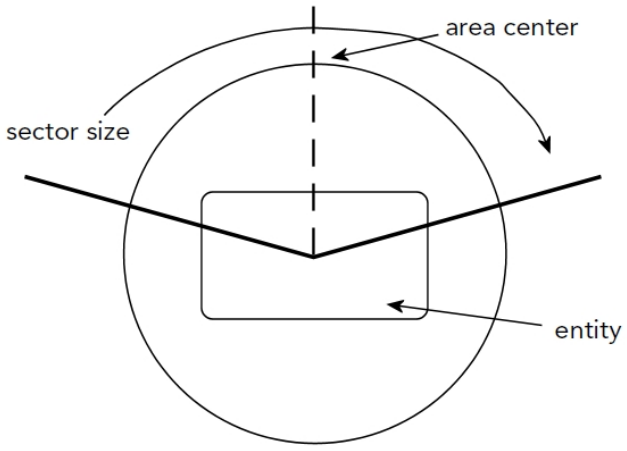
40.71. Sector of Responsibility

A simulation object's sector of responsibility is used with detection tables to determine the priority of targets. Normally, the priority of targets that are outside a simulation object's sector of responsibility is lower than the priority of targets in its sector of responsibility. For information about the target detection tables, see "76.2. Detection Tables" on page 1516.

You must specify a sector center and a sector size. Sector center is the center of the area you wish to designate, and sector size is the size of the area, both in degrees. Degrees are measured as 0 to the front (based on the heading), increasing clockwise. If the Turret Scan Controller is mounted on an entity with an articulating turret, then the turret scans the sector of responsibility specified. Sector center is the midpoint of the area. Figure 299 illustrates these parameters. The sector size is approximately 140 degrees, a bit less than half the circle. The sector center is at 0 degrees.

ⓘ The default sector of responsibility is 90 degrees, centered at zero degrees.

Figure 299. Sector of responsibility parameters



➤ **To set an object's sector of responsibility:**

1. Select the simulation object.
2. Choose **Set > Engagement > Sector of Responsibility**. The Sector of Responsibility dialog box opens.
3. In the Size text box, set the size of the area to be scanned, in degrees.
4. In the Center text box, set the center of the scan area, in degrees.
5. Click OK.

40.72. Sensor Aim

The Sensor Aim set data request specifies where a gimbaled sensor should look.

You can also aim a gimbaled sensor from the Sensor View Control Panel. For details, see "62.4.2 Controlling the Sensor View" on page 1230.

➤ **To aim a sensor:**

1. Select an entity that has a gimbaled visual sensor.
2. Choose **Set > Sensors > Sensor Aim**. The Sensor Aim dialog box opens.
3. Choose an aiming option and provide the required input:
   - Aim at Point. Click on the terrain or type in coordinates. By default, the altitude is the altitude of the entity.
   - Aim at Object. Select the simulation object to aim at.
   - Aim at Angle. Specify the Azimuth (the horizontal angle relative to the sensing entity) and elevation (the vertical angle that the sensor aims at).
   - Scan. By default, the sensor scans back and forth within a 180 degree range. You can change the elevation and azimuth angles in the gimbaled sensor system definition file.
4. Click OK.

40.73. Sensor Field of View Color

The Sensor Field of View Color set data request lets you override the color that VR-Forces automatically assigns to the field of view.

➤ **To specify the color of a sensor field of view:**

1. Select an entity that has a sensor.
2. Choose **Set > Sensors > Sensor Field of View Color**. The Sensor Field of View Color dialog box opens.
3. In the Sensor list, select the sensor you want to specify the color of.
4. Click the color swatch. A color picker dialog box opens.
5. Select the color you want.
6. Click OK to close the color picker.
7. Click OK.

40.74. Sensor Zoom

The Sensor Zoom set data request lets you change the field of view of a gimbaled visual sensor, essentially zooming in on the aim point.

You can also zoom a gimbaled sensor from the Sensor View Control Panel. For details, see "62.4.2 Controlling the Sensor View" on page 1230.

➤ **To zoom a visual sensor:**

1. Select the entity whose sensor you want to zoom.
2. Choose **Set > Sensors > Sensor Zoom**. The Sensor Zoom dialog box opens.
3. If the entity has more than one sensor, in the Sensor list, select the sensor you want to zoom.
4. Type the zoom level you want or drag the slider.
5. Click OK.

40.75. Skill Level

The Skill Level set data request allows you to specify an entity's level of skill with hand-held or ballistic weapons systems. There are three levels: Novice, Experienced, and Expert. The skill level is a modifier on the probability of a hit for hand-held or ballistic weapons. On indirect fire, it acts as a modifier on the error of the shot. The default skill level for all simulation objects is Expert.

Each weapon system has a configurable maximum effect that the skill level can have on its probability. For example, a weapon with built-in corrections would not be affected as much as a hand-held M16. The skill level is a percentage modifier that is applied to the probability to hit. The maximum effect can be set differently for different weapon systems in the Simulation Object Editor. If the maximum effect is set to .20 (20%), then the probability can shift as much as 20% for a novice, 10% for an experienced entity, and 0% for an expert.

In addition, the modifier affects the three positional errors on indirect artillery and mortars, which are also set in the Simulation Object Editor. The error of indirect fire is set to be close to standard error data points. You can also edit the object to vary the error of indirect fire based on the range at which the munition is being fired.

➤ **To set an entity's skill level:**

1. Select the entity whose skill level you want to adjust.
2. Choose **Set > Disposition > Skill Level**. The Skill Level dialog box opens.
3. In the Skill Level list, select the level that you want the entity to have.
4. Click OK.

40.76. Sonar Depth

The Sonar Depth set data request lets you set the depth at which an entity's sonar operates. This allows you to simulate a helicopter dipping a sonar sensor into the water or a vessel dipping a sensor to a lower depth. Since sonar only works when the sensor is in the water, setting the sonar depth is the only way for helicopter-born sonar to work.

The sonar stays at this depth regardless of the altitude or depth of the entity. If you set the depth to be the depth of the entity with the Use Entity Depth option, it has the effect of retracting the sonar. If you set the sonar to the depth of the entity and the entity changes depth, the sonar stays at the assigned depth.

➤ **To set sonar depth:**

1. Select the entity.
2. Choose **Set > Sensors > Sonar Depth**. The Sonar Depth dialog box opens.
3. Specify the depth.
4. Optionally, select Use Entity Depth to set a dipped sonar to the depth of the entity.
5. Click OK.

40.77. Spot Reports

A spot report is a report by a simulation object to members of its force that it has detected a member of another force. You can use spot reports for display only (meaning you see spot-reported entities but members of your force are not aware of them), or for avoidance and targeting (meaning members of your force can "see" them too).

For display purposes, you can set spot reports on or off for all simulation objects (global setting) from the Spot Report Settings page of the Application Settings dialog box. For more information about spot reports, including setting them globally, see "9.1. Displaying Spot-Reported Simulation Objects" on page 290.

For avoidance and targeting purposes, you must enable the display of spot-reported simulation objects as described in "Displaying Spot-Reported Simulation Objects", and also set each simulation object you want to avoid/target either to broadcast spot reports or to send spot reports to particular simulation objects in your force, as described here. That is to say, you must set entities of Force A to send spot reports about Force B to allow Force A's entities to avoid/target the entities of Force B

> ⓘ You cannot set a unit in an entity-level scenario to send spot reports; you can only set spot reports for individual simulation objects. However, you can select multiple individual simulation objects and set spot reports for all of them at the same time.

➤ **To set spot reports on or off:**

1. Select the simulation object.
2. Choose **Set > Sensors > Spot Reports**. The Spot Reports dialog box opens. If you are turning spot reports off, skip to step 5.
3. Select one of these options:
   - Only to GUI. Send spot reports only to the GUI. Do not send spot reports to other sim engines or to other simulation objects. This option can improve performance. However, since entities do not receive the spot reports, they cannot use that information for targeting purposes. Choose this option if you simply want to display the objects sensed by this entity.
   - Broadcast. Send spot reports to all simulation objects in the same unit as the entity sending the spot report. (This is based on the behavior of radio messages because spot reports are transmitted as radio messages.)

- Send to Specific Entities. Send spot reports to simulation objects that you specify.
- Superiors and Peers. Send spot reports to simulation objects in the same unit.

4. If you selected Send to Specific Entities, the simulation object list is enabled. Select the simulation objects to which you want this simulation object to send spot reports.
5. Select an option from the Spot Reports Enabled list. If you choose On or Off, the setting persists regardless of the global spot report setting. If you want the simulation object to use the global spot report setting, choose Use Global Setting.
6. Click OK.

40.78. Stow Hand Item

The Stow Hand Item set data request hides or displays a hand item for a human character.

➤ **To stow or display a hand item:**

1. Select the entity.
2. Choose **Set > Appearance > Stow Hand Item**. The Stow Hand Item dialog box opens.
3. In the Stow Item list, select Yes or No.
4. Click OK.

40.79. Subsystem Enabled

The Subsystem Enabled set data request enables or disables one of the entity's subsystems. Only certain subsystems can be enabled or disabled, such as sensors and weapons.

➤ **To enable or disable a subsystem:**

1. Select the entity.
2. Choose **Set > Subsystems > Subsystem Enabled**. The System Enabled dialog box opens.
3. In the subsystem list, select the subsystem that you want to enable or disable, or select All.
4. In the Enabled list, select Yes or No.
5. Click OK.

40.80. Superior

The Superior set data request lets you add a simulation object (individual or unit) to a unit or move a simulation object from a unit to the force level. The superior unit must already exist. For aggregate-level scenarios, the superior must be a higher echelon unit.

➤ **To specify an object's superior:**

1. Select the simulation object.
2. Choose **Set > Other > Superior**. The Superior dialog box opens.
3. Optionally, filter the list.
4. Select the unit that you want to be this simulation object's superior. If you want to move the simulation object out of a unit to the force level, select the Set Force As Superior check box.
5. Click OK.

40.81. Surface Vessel Lighting

The Surface Vessel Lighting set data request causes a watercraft to publish its lighting information as specified. The 3D visuals on the ship model reflect the lighting if the model is configured with those lights.

➤ **To set the lights on a surface vessel:**

1. Select the entity.
2. Choose **Set > Appearance > Surface Vessel Lighting**. The Surface Vessel Lighting parameters display.
3. Specify the settings for the lights.
4. Click OK.

40.82. Surrendered

The Surrendered set data request only applies to lifeforms. When a lifeform is in the surrendered state, it does not fire at other simulation objects and other simulation objects do not fire at it.

➤ **To set an entity to be surrendered:**

1. Select the entity.
2. Choose **Set > Disposition > Surrendered**. The Surrendered dialog box opens.
3. Select an option from the list.
4. Click OK.

40.83. Synchronize Laser Code

The Synchronize Laser Code set data request lets an entity with laser-guided weapons easily synchronize its laser code with the laser code of the lasing entity. The alternative to this command would be to use the Laser Code set data request to set the code for the lasing entity and the firing entity.

➤ **To synchronize laser codes:**

1. Select the entity that has a laser-guided weapon.
2. Choose **Set > Laser Designator > Synchronize Laser Code**. The Synchronize Laser Code dialog box opens.
3. Select the lasing entity that you want this entity to synchronize with.
4. Click OK. The laser code for the selected entity is changed to match that of the lasing entity.

40.84. TACAN Transmitter

The Set TACAN Transmitter set data request allows you to set fixed-wing aircraft and TACAN Ground Station entities to publish tactical air navigation aids. These are published on the network as a type of radio transmission.

➤ **To set the TACAN transmitter:**

1. Select the entity.
2. Choose **Set > Sensors > Set TACAN Transmitter**. The Set TACAN Transmitter dialog box opens.
3. From the Operational Status list, choose either On or Off.
4. Enter an identification code.
5. Enter a channel number between 1 and 126.
6. In the list next to Channel, select X or Y to indicate the pulse band.
7. Click OK.

40.85. Tanker Lighting

The Tanker Lighting set data request causes an air tanker to publish its lighting information as specified. The 3D visuals on the fixed wing model reflect the lighting if the model is configured with those lights.

There is also a generic lighting set data request for all fixed-wing aircraft. See "40.6. Aircraft Lighting" on page 868.

➤ **To set the lights on a air tanker:**

1. Select the entity.
2. Choose **Set > Appearance > Tanker Lighting**. The Tanker Lighting dialog box opens.
3. Specify the settings for the lights.
4. Click OK.

40.86. Target

You can command an entity to specifically target another entity. For this set data request to be effective, the target must be visible to the targeting entity. When you set a target, you override the priorities set by the target selection controller. If the specified target is in the list of candidate targets, it gets the highest priority as a target. However, if the specified target is not visible as a target to the targeting entity, the entity fires at the other available targets. It does not keep looking for the specified target.

➤ **To set an entity's target:**

1. Select the entity.
2. Choose **Set > Engagement > Target**. The Target dialog box opens.
3. Optionally, filter the entity selection list.
4. In the Target list, or on the terrain, select the entity to target.
5. Click OK.

40.87. Target Priority List - Add

The Target Priority List - Add set data request allows you to set a simulation object's priority for targeting other entities in the scenario. You can specify that the simulation object should only target the entities in the priority list using the Rules of Engagement set data request as described in "40.69. Rules of Engagement" on page 896. To remove all items from a target priority list, use the Clear Target Priority List set data request; see "40.88. Target Priority List - Clear" on the next page for more information.

➤ **To add a target to the priority list:**

1. Select the simulation object.
2. Choose **Set > Engagement > Target Priority List - Add**. The Target Priority list dialog box opens.
3. From the Priority list, select High, Medium, or Low.
4. Optionally, filter the entity selection list. The default is Opposing Entities unless there are no opposing entities in the scenario.
5. In the Target list or on the terrain, select the entity to target.
6. Click OK.

40.87.1 Viewing the Target Priority List

You can view an entity's list of prioritized targets using the viewing the State Data page of the Information dialog box or the Target Priority List -Update set data request. (For the latter, see "40.89. Target Priority List - Update" on the next page.)

➤ **To view the current target priority list in the State Data page:**

1. Select the entity.
2. Choose **Objects > Information**.

3. Select the State Data page.
4. Scroll to the Target Priority List line.

40.88. Target Priority List - Clear

Use the Target Priority List - Clear set data request to remove all targets from an entity's priority list. This applies to all targets that were added to or updated in the target priority list. See "40.87. Target Priority List - Add" on the previous page and "40.89. Target Priority List - Update" below.

➤ To remove entities from a target priority list, select the entity and then choose **Set > Engagement > Target Priority List - Clear**.

40.89. Target Priority List - Update

The Target Priority List - Update set data request allows you to add and remove targets from an entity's priority list, as well as change the priority for each target.

ⓘ You cannot use the Target Priority List - Update set data request in plans. Use it to manage an entity's target priority list outside of plans.

➤ **To update an entity's target priority list:**

1. Select the entity.
2. Choose **Set > Engagement > Target Priority List - Update**. The Update Target Priority List dialog box opens.
3. Manage the list of targets:
   - To add a target to the list, click the Add button (+). In the Add to Target Priority List dialog box, select the priority and the target, and then click OK.
   - To remove a target from the list, select the row and then click the Delete button (🗑).
   - To change a target's priority, select High, Medium, or Low from that target's Priority list.
4. Click OK.

40.90. Tasked by Superior

You can order a simulation object that is part of a unit, and which is executing its own plan or is conducting an independent task, to stop executing those tasks and take commands from its unit. When you order a simulation object to be tasked by its superior, it stops conducting independent tasks and waits for the next order from the unit. (It does not execute the unit's current task, if any.) For more information about being tasked by superior, see "29.4.2 Independently Tasking Unit Members" on page 572.

➤ **To order an object to be tasked by its superior:**

1. Select the simulation object.
2. Choose **Set > Other > Tasked by Superior**.

40.91. TDL Messaging

The TDL Messaging set data request enables or disables TDL messages for a simulation object or sets it to use the global setting.

➤ **To specify TDL messaging:**

1. Select the simulation object.
2. Choose **Set > Other > TDL Messaging**. The TDL Messaging dialog box opens.
3. Select an option from the TDL Messaging Enabled list.
4. Click OK.

40.92. Tracer Use

You can specify use of tracer rounds for entities that have weapons that support them. To see tracers, you must enable them for the observer you are using. They are not supported in 2D observer modes.

➤ **To enable or disable use of tracers:**

1. Select the entity for which you want to enable or disable use of tracers.
2. Choose **Set > Engagement > Tracer Use**. The Tracer Use dialog box opens.
3. Choose an option from the Use Tracers list.
4. Click OK.

40.93. Undetectable

The Undetectable set data request sets any simulation object to be undetectable by other objects. The undetectable, or stealth state, has these characteristics:

- Objects in stealth mode can take damage. They are not invulnerable. This means that missiles/bullets can still impact with stealth mode entities. Missiles already launched at an entity while it is not in stealth mode will continue to track an entity even after it is turned to stealth mode, and may hit and destroy it.
- Stealth mode entities can trigger mines.
- Lua scripts using the getObjectsNear() function will find stealth objects in the list, but may check the "stealth" property of the object if the particular script should ignore stealth entities.
- Collision avoidance takes into account stealth entities.
- Stealth entities block line of sight checks that pass through them, just as they would if they were not stealth.

➤ **To set a simulation object to be undetectable:**

1. Select the entity you want to be undetectable.
2. Choose **Set > Disposition > Undetectable**. The Undetectable dialog box opens.
3. Select an option in the Undetectable list.
4. Click OK.

40.94. Unit Patches

For soldier entities, you can set unit patches to display. At this time, not all soldiers are configured with units.

The Apply Selected Patches Immediately check box allows you to specify whether you want each patch to update when you select it. This is selected by default. If you clear the check box, the unit patches change after you click OK.

You can also specify a name and rank, as described in "40.52. Name Tag" on page 889 and "40.63. Rank Patch" on page 894.

➤ **To specify a soldier's unit patches:**

1. Select the soldier entity.
2. Choose **Set > Appearance > Unit Patches**. The Unit Patches parameters display.
3. Optionally, clear the Apply Selected Patches Immediately check box.
4. Select the desired unit from each list (primary and secondary).
5. Click OK.

40.95. Urgency (Human)

The Urgency set data request controls the speed at which a human entity performs animations, such as transitions between postures.

➤ **To set a human's urgency:**

1. Select the entity.
2. Choose **Set > Action > Urgency**. The Urgency dialog box opens.
3. Use the Urgency list to specify the level of urgency, from Slowest to Immediate. Immediate skips any transition animations. The other urgency settings move through the animations at various speeds.

40.96. Set Data Requests for Units

Some set data requests are specifically for use by units.

40.96.1 Assign Transport

Use the Assign Transport set data request to designate a vehicle for a squad or team to use for mount and dismount tasks. You can also clear a previously designated vehicle from a unit.

➤ **To set the vehicle for a squad or team:**

1. Select the unit.
2. Choose **Set > Embarkation > Assign Transport**.
3. Select the ground entity.
4. If the unit was previous assigned to a designated vehicle and you do not want it to have that designation anymore, select Clear Assignment.
5. If you want the assignment (or the clearing of the assignment) to apply to any of its subordinate units, select Apply to Subordinate Units.
6. Click OK.

40.96.2 Dismounted

The Dismounted set data request immediately dismounts a platoon, squad, or team from a vehicle (as opposed to simulating the unit dismount, as done by the Dismount Platoon, Dismount Squad, or Dismount Team tasks).

➤ **To set a platoon, squad, or team to be dismounted:**

1. Select the unit.
2. Choose **Set > Embarkation > Dismounted**. The unit becomes dismounted from the vehicle.

40.96.3 Dismounted Headquarters Section

The Dismounted Headquarters Section set data request immediately dismounts the members of a headquarters section from the vehicle or vehicles they are mounted on (as opposed to simulating the HQ section dismount, as done by the Dismount Headquarters task). Optionally, you can specify whether the platoon sergeant or other habitually mounted entities remain on board.

➤ **To set a headquarters section to be dismounted:**

1. Select the headquarters section.
2. Choose **Set > Embarkation > Dismounted Headquarters Section**. The Dismounted Headquarters Section dialog box opens.
3. Optionally, select or clear the Retain Habitual Mountees check box.
4. Click OK. The unit becomes dismounted from the vehicle.

40.96.4 Lock Embarkation

Use the Lock Embarkation set data request to instruct a vehicle crew to ignore mount or dismount tasks. You can also clear a previous lock.

➤ **To lock or unlock a vehicle crew response to mount and dismount tasks:**

1. Select the crew and then choose **Set > Embarkation > Lock Embarkation**. The Lock Embarkation dialog box opens.
2. Select or clear the lock check box.
3. Click OK.

40.96.5 Mounted

The Mounted set data request immediately mounts the subordinates of a platoon, squad, or team on a vehicle (as opposed to simulating the unit mount, as done by the Mount Platoon, Mount Squad, or Mount Team tasks). The vehicle they mount is the one specified by the Assign Transport set data request. See "40.96.1 Assign Transport" on the previous page.

➤ **To set a platoon, squad, or team to be mounted:**

1. Select the unit.
2. Choose **Set > Embarkation > Mounted**. The unit becomes mounted on the assigned vehicle.

40.96.6 Mounted Crew

The Mounted Crew set data request immediately mounts a vehicle crew on its assigned transport (as opposed to simulating the unit mounting the vehicle, as done by one of the Mount tasks). The vehicle they mount is the one specified by the Assign Transport set data request. See "40.96.1 Assign Transport" on the previous page.

➤ **To set a vehicle crew to be mounted:**

1. Select the crew.
2. Choose **Set > Embarkation > Mounted Crew**. The crew becomes mounted on the assigned vehicle.

40.96.7 Mounted Headquarters Section

The Mounted Headquarters Section set data request immediately mounts the members of a headquarters section on one or more vehicles (as opposed to simulating the mount, as done by the Mount Headquarters task). The Platoon Leader and Radio Telephone Operator mount the first vehicle you specify, and the Platoon Sergeant and Forward Observer mount the second.

➤ **To set an headquarters section to be mounted:**

1. Select the HQ section.
2. Choose **Set > Embarkation > Mounted Headquarters Section**. The Mounted Headquarters Section dialog box opens.
3. Select the vehicles in the Objects List or on the terrain. They are added to the Transports list.

40.96.8 Ordered Formation

Use the Ordered Formation set data request to specify the formation that the unit should assume while moving.

This does not change their formation when a unit is not in motion. If you want a unit that is currently stationary to change its formation, see "30.26. Move Into Formation" on page 603.

➤ **To set the ordered formation of unit:**

1. Select the unit.
2. Choose **Set > Action > Ordered Formation**. The Set Ordered Formation parameters display.
3. Select the formation that you want the unit to assume.
4. Click OK.

40.96.9 Unloaded

The Unloaded set data request causes a vehicle to immediately dismount its passengers. This applies to both mounted and embarked passengers.

➤ **To unload passengers from a vehicle:**

1. Select the vehicle.
2. Choose **Set > Embarkation > Unloaded**. The passengers are immediately dismounted from the vehicle.



41. Setting the State of Aggregate-Level Simulation Objects

41.1. Activity 915
41.2. Ammunition 915
41.3. Ammunition Pacing/Tracking 915
41.4. Assign Aircraft 916
41.4.1 Editing Aircraft Assignments 917
41.4.2 Deleting Aircraft Assignments 917
41.5. Assume Unknown is Hostile 918
41.6. Capabilities 918
41.7. Concealed 918
41.8. Contamination 918
41.9. Current Speed 918
41.10. Destroyed 918
41.11. Disembarked 918
41.12. Embarked 918
41.13. Emitter 918
41.14. Engagement Zones 918
41.15. Engagement Results 919
41.16. Equipment Pacing/Tracking 919
41.17. Facilities 919
41.18. Food, Water, Fuel, Oil and Lubricant 920
41.19. Food, Water, Fuel, Oil and Lubricant Pacing/Tracking 921
41.20. Force 921
41.21. Has Drag Chute 921
41.22. Heading 921
41.23. Health 921
41.24. IFF 922
41.25. Loadout 922
41.26. Location 922

41.27. MOPP Level 922
41.28. Morale 923
41.29. Notify Level 923
41.30. Ordered Speed 923
41.31. Percent Complete 923
41.32. Personnel and Supplies 923
41.33. Posture (Unit) 924
41.34. Preferred Targets 924
41.35. Radar Mode 924
41.36. Reorganize 924
41.37. Resources 924
41.38. Resources Pacing/Tracking 925
41.39. Restore 925
41.40. Resupply Mode 925
41.41. Role 926
41.42. Rules of Engagement 926
41.43. Sector of Responsibility 926
41.44. Spot Reports 926
41.45. Subsystem Enabled 926
41.46. Superior 926
41.47. Supplying 926
41.48. Tasked by Superior 927
41.49. TDL Messaging 927
41.50. Weapons Pacing/Tracking 927

41.1. Activity

You can give a simulation object an activity. Activities are configured by simulation object type, so they may vary from one simulation object to another. Some of the available activities include:

- Unknown
- Attacking
- Bivouacking
- Defending
- Emplacing Obstacle
- Indirect fire
- Jamming
- Other
- Patrolling
- Probing
- Retreating

➤ **To specify an activity:**

1. Select the simulation object.
2. Choose **Set > Disposition > Activity**. The Activity dialog box opens.
3. Select an activity from the list.
4. Click OK.

41.2. Ammunition

You can specify the amount of each type of ammunition that a simulation object has.

➤ **To set the ammunition level:**

1. Select the simulation object.
2. Choose **Set > Disposition > Ammunition**. The Ammunition dialog box opens.
3. Select the check box for each type of ammunition that you want to set.
4. For each selected type of ammunition, adjust the slider or type a value.
5. Click OK.

41.3. Ammunition Pacing/Tracking

Ammunition Pacing/Tracking lets you specify that a particular ammunition type be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

➤ **To specify pacing or tracking for a unit's ammunition:**

1. Select the simulation object.
2. Choose **Set > Disposition > Ammunition Pacing/Tracking**. The Ammunition Pacing/Tracking dialog box opens.
3. Select the check box for each type of ammunition for which you want to specify pacing or tracking.
4. For each selected ammunition type, select an option from the list.
5. Click OK.

41.4. Assign Aircraft

The Assign Aircraft set data request lets you assign aircraft to an air base, edit existing assignments, and remove aircraft from the air base.

The aircraft assignments process works differently depending on whether you are using it directly from the Set menu or as part of a plan. When you assign aircraft from the Set menu, the Assign Aircraft shows all of the aircraft types currently assigned to the air base and the amount of each type of aircraft. By default (the Set Assigned Aircraft to This List option), when you click OK the new or updated list of assigned aircraft replaces whatever the previous list was. This is the way set data requests typically work.

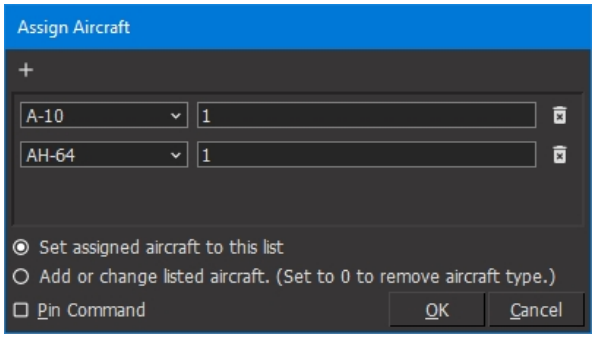
If you include an Assign Aircraft set data request in a plan, there is no way to know what the state of aircraft assignments will be when the plan statement is executed. Therefore, the default behavior for Assign Aircraft when used in a plan is to add or change only the specific aircraft assignments listed in the Assign Aircraft dialog box, rather than resetting all aircraft assignments (the Add or Change Listed Aircraft option).

The Assign Aircraft dialog box has a set of option buttons whose default settings support the different use cases described in previous paragraphs. However, you can choose the alternative options if you want to.

➤ **To add an aircraft type to an air base:**

1. Select the air base.
2. Choose **Set > Air Base > Assign Aircraft**. The Assign Aircraft parameters display (Figure 300). If the air base already has aircraft assigned to it, they are listed in the window along with the number of each type assigned. (If you are editing a plan, the current aircraft assignments are not listed.)

Figure 300. Assign Aircraft dialog box



3. For each aircraft type that you want to assign to the air base:
    a. Click the Add button (+). A row is added to the dialog box window.
    b. In the aircraft list, select the type of aircraft you want to add.
    c. In the text box, type the number of this type of aircraft you want to add.
4. Optionally, edit any existing aircraft assignments.
5. Select the appropriate option button. (It is usually best to accept the default option.)
6. Click OK.

41.4.1 Editing Aircraft Assignments

You can add or remove aircraft types for an air base and you can change the number of aircraft of a specific type. When you add aircraft, they are added in the Available state. No fuel, munitions, or air crews are added.

If you reduce the number of aircraft of a specific type, the aircraft are taken from the Available aircraft first, followed by In-Repair and Assigned. If Assigned aircraft are removed, their fuel, munitions, and air crews are returned to the air base.

41.4.2 Deleting Aircraft Assignments

➤ **To delete an aircraft assignment for an air base:**

1. Select the air base.
2. Choose **Set > Air Base > Assign Aircraft**. The Assign Aircraft dialog box opens. If the air base already has aircraft assigned to it, they are listed in the window along with the number of each type assigned. (If you are editing a plan, the current aircraft assignments are not listed.)
3. If you are not editing a plan, click the Delete button next to the row that you want to delete or set the number of aircraft to zero. If you are editing a plan, add the aircraft type that you want to delete and set its amount to 0.
4. Select the appropriate option button. (It is usually best to accept the default option.)
5. Click OK.

41.5. Assume Unknown is Hostile

For details, please see "40.11. Assume Unknown is Hostile" on page 870.

41.6. Capabilities

For details, please see "40.17. Capabilities" on page 873.

41.7. Concealed

For details, please see "40.19. Concealed" on page 873.

41.8. Contamination

The Contamination set data request lets you set the contamination type for a simulation object.

➤ **To set contamination:**

1. Select the simulation object.
2. Choose **Set > Disposition > Contamination**. The Contamination dialog box opens.
3. Select a contamination type from the list.
4. Click OK.

41.9. Current Speed

For details, please see "40.21. Current Speed" on page 874.

41.10. Destroyed

For details, please see "40.23. Destroyed" on page 875.

41.11. Disembarked

For details, please see "40.27. Disembarked" on page 877.

41.12. Embarked

For details, please see "40.28. Embarked" on page 877.

41.13. Emitter

For details, please see "40.30. Emitter" on page 878.

41.14. Engagement Zones

For details, please see "40.31. Engagement Zones" on page 879.

41.15. Engagement Results

An engagement result is a text string included in entity state update messages that are sent over the network. The purpose is for a human participant to comment on the current state of a simulation object as a simulation progresses. Once specified, the text string continues to be sent until the expiration time is reached or a new engagement result is set.

➤ **To set engagement results:**

1. Select the simulation object.
2. Choose **Set > Engagement > Engagement Results**. The Engagement Results dialog box opens.
3. Specify the engagement result. This can be any text you want to enter.
4. Set the expiration time in days and hours:minutes:seconds.
5. Click OK.

41.16. Equipment Pacing/Tracking

Equipment Pacing/Tracking lets you specify that a particular type of equipment be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

➤ **To specify pacing or tracking for a unit's equipment:**

1. Select the simulation object.
2. Choose **Set > Disposition > Equipment Pacing/Tracking**. The Equipment Pacing/Tracking dialog box opens.
3. Select the check box for each type of equipment for which you want to specify pacing or tracking.
4. For each selected ammunition type, select an option from the list.
5. Click OK.

41.17. Facilities

Use the Facilities set data request to specify all of the capabilities of an air base:

- Runways. Specify a name, length, width, and main direction for each runway.
- Has Night Support. Able or not able to launch missions at night.
- ORP Aircraft Capacity. The number of units that can be in operational readiness platform parking spaces.
- Apron Area. The size of the apron. VR-Forces calculates the number of aircraft that can be on the apron based on their footprints.

- Repair Hangar Aircraft Capacity. The number of aircraft that can be in the repair hangar at one time.
- Aviation Fuel Capacity. The amount of aviation fuel the air base can store.
- Munition Storage Capacity. The amount of munitions the air base can store.
- Barracks Capacity. The number of personnel that can be on the air base and in the barracks.
- Is Apron Concealed.
- Is Hangar Concealed.

➤ **To specify air base facilities:**

1. Select an air base.
2. Choose **Set > Air Base > Facilities**. The Facilities dialog box opens.
3. To add a runway:
    a. Click the Add button (+). A new runway is added to the runway list.
    b. Type values in each column of the runway description.
4. To delete a runway:
    a. Put the cursor in a cell in a runway row.
    b. Click the Delete button ().
5. To enable night support, select Yes in the list.
6. For the storage capacities, enter the amount of that item that you want to store in the air base.
7. To revert any settings or the configured runways to their original values, click the Revert button () for that setting.
8. To set the concealment status of the apron or hangar, select an option in the appropriate list.
9. Click OK.

41.18. Food, Water, Fuel, Oil and Lubricant

You can set the resources for a simulation object.

➤ **To set the level of food, water, fuels, and lubricants:**

1. Select the simulation object.
2. Choose **Set > Disposition > Food, Water, Fuel, Oil and Lubricant**. The Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant dialog box opens.
3. Select the check box for each type of resource you want to specify.
4. For each selected resource type a value.
5. Click OK.

41.19. Food, Water, Fuel, Oil and Lubricant Pacing/Tracking

Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking lets you specify that a particular resource be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

➤ **To set pacing and tracking for food, water, fuel, and lubricants:**

1. Select the simulation object.
2. Choose **Set > Disposition > Food, Water, Fuel, Oil and Lubricant Pacing/Tracking**. The Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant Pacing/Tracking dialog box opens.
3. Select the check box for each type of resource for which you want to specify pacing or tracking.
4. For each selected resource select an option from the list.
5. Click OK.

41.20. Force

For details, please see "40.32. Force" on page 879.

41.21. Has Drag Chute

The Has Drag Chute set data request specifies whether the aircraft in an air unit have drag chutes.

➤ **To set a drag chute:**

1. Select an aircraft unit.
2. Choose **Set > Disposition > Has Drag Chute**. The Has Drag Chute dialog box opens.
3. Select an option from the Has Drag Chute list.
4. Click OK.

41.22. Heading

For details, please see "40.39. Heading" on page 882.

41.23. Health

Health is a value based on a simulation object's total personnel, equipment, and weapons. The Health set data request lets you restore personnel to their full values or set the overall health of the simulation object. For details about simulation object health, see "27.1. The Aggregate Warfare Model" on page 528.

➤ **To set an object's health:**

1. Select the simulation object.
2. Choose **Set > Disposition > Health**. The Health dialog box opens. It displays the current state of personnel, equipment, and weapons.
3. To restore all personnel, select Reset Kill, Captured, Wounded, Missing.
4. To specify the overall health of the simulation object, adjust the Health slider or type a number in the Health box. As you move the slider to the left, the amount of personnel, equipment, and weapons decreases.
5. Click OK.

41.24. IFF

For details, please see "40.41. IFF" on page 883.

41.25. Loadout

The Loadout set data request lets you specify the loadout for an air unit. It lists the contents of the selected loadout.

➤ **To set the loadout for an air unit:**

1. Select an air unit.
2. Choose **Set > Disposition > Loadout**. The Loadout dialog box opens.
3. Select an option from the Loadout Name list. The number of munitions available in the loadout displays.
4. Click OK.

41.26. Location

For details, please see "40.49. Location" on page 887.

41.27. MOPP Level

When you set a unit's MOPP level, the MOPP level changes instantly. To change MOPP level over time, and for details on how MOPP levels work, see the Change MOPP Level task in "35.5.1 Change MOPP Level" on page 732.

➤ **To set the MOPP level:**

1. Select the simulation object.
2. Choose **Set > Disposition > MOPP Level**. The MOPP Level dialog box opens.
3. Select an option from the list.
4. Click OK.

41.28. Morale

➤ **To set a unit's morale:**

1. Select the simulation object.
2. Choose **Set > Disposition > Morale**. The Morale dialog box opens.
3. Adjust the slider or type a value from 0 through 100.
4. Click OK.

41.29. Notify Level

For details, please see "40.55. Notify Level" on page 890.

41.30. Ordered Speed

For details, please see "40.57. Ordered Speed" on page 891.

41.31. Percent Complete

Percent Complete lets you set the degree of completion for combat engineering objects.

➤ **To set the percent complete:**

1. Select a combat engineering object.
2. Choose **Set > Disposition > Percent Complete.** The Percent Complete dialog box opens.
3. Adjust the slider or type a value in the box.
4. Click OK.

41.32. Personnel and Supplies

The Personnel and Supplies set data request lets you specify the number of air crews and ground crew available to an air base. You can also set the amount of munitions and aviation fuel. The total of air crews and ground crew cannot exceed the barracks capacity, which is set using the Facilities set data request. You also use the Facilities set data request to specify the maximum amount of munitions and aviation fuel that an air base can have.

➤ **To specify personnel and supplies for an air base:**

1. Select an air base.
2. Choose **Set > Air Base > Personnel and Supplies**. The Personnel and Supplies dialog box opens.
3. Optionally, type a value for Ground Crew Available.
4. Optionally, type a value for Air Crew Available.
5. Optionally, type a value for the amount of munitions.

6. Optionally, type a value for the amount of aviation fuel.
7. To revert any setting to its original value, click the Revert button () for that setting.
8. Click OK.

41.33. Posture (Unit)

You can set the posture of a unit. The posture changes instantly. To change posture over time, use the Change Posture task.

➤ **To set an object's posture in an aggregate-level scenario:**

1. Select the simulation object.
2. Choose **Set > Disposition > Posture**. The Posture dialog box opens.
3. Select an option from the list.
4. Click OK.

41.34. Preferred Targets

Units can target up to three simulation objects. By default, they target the three simulation objects that are most vulnerable to attack. If you want a simulation object to target specific simulation objects that it might not choose automatically, the Preferred Targets set data request lets you specify them as preferred targets.

➤ **To specify preferred targets:**

1. Select a unit.
2. Choose **Set > Engagement > Preferred Targets**. The Preferred Targets dialog box opens.
3. Select up to three simulation objects.
4. Click OK.

41.35. Radar Mode

For details, please see "40.62. Radar Mode" on page 894.

41.36. Reorganize

For details, please see "40.66. Reorganize" on page 895.

The change takes place immediately. For more information about reorganization, please see "25.1.1 Reorganizing Units" on page 516.

41.37. Resources

For details, please see "40.67. Resources" on page 896.

41.38. Resources Pacing/Tracking

Resources Pacing/Tracking lets you specify that a resource type be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

➤ **To set pacing and tracking for a resource:**

1. Select the simulation object.
2. Choose **Set > Disposition > Resources Pacing/Tracking**. The Resources Pacing/Tracking dialog box opens.
3. Select Resources.
4. Click OK.

41.39. Restore

For details, please see "40.68. Restore" on page 896.

41.40. Resupply Mode

Resupply Mode specifies one of these resupply modes for simulation objects:

- None. The simulation object does not resupply itself.
- From Supply Units. When a resupply unit is in range, the unit receives supplies from it.
- From Self-Continuous. The unit resupplies itself continuously based on its resupply rate parameters. For details, see "27.3. Logistics" on page 537.
- From Self-Periodic. The unit periodically resets all of its supplies, except health, to the base level. This level is reduced by health losses.

➤ **To specify the resupply mode:**

1. Select the unit.
2. Choose **Set > Logistics > Resupply Mode**. The Resupply Mode dialog box opens.
3. Select a resupply mode from the Resupply Mode list.
4. If you select From Self-Periodic, specify the resupply period in the Periodic Resupply Period boxes.
5. Click OK.

41.41. Role

The Role set data request lets you specify the role of an air unit, such as fighter, attack, bomber, or QRA.

➤ **To set the role of an air unit:**

1. Select an air unit.
2. Choose **Set > Disposition > Role**. The Role dialog box opens.
3. Select an option in the Role list.
4. Click OK.

41.42. Rules of Engagement

For details, please see "40.69. Rules of Engagement" on page 896.

41.43. Sector of Responsibility

For details, please see "40.71. Sector of Responsibility" on page 898.

41.44. Spot Reports

For details, please see "40.77. Spot Reports" on page 901.

41.45. Subsystem Enabled

For details, please see "40.79. Subsystem Enabled" on page 902.

41.46. Superior

For details, please see "40.80. Superior" on page 902.

41.47. Supplying

The Supplying set data request sets a supply unit to stop providing supplies or to start providing supplies.

➤ **To stop or start a supply unit providing supplies:**

1. Choose **Set > Logistics > Supplying**. The Supplying dialog box opens.
2. Select Stop or Start.
3. Click OK.

41.48. Tasked by Superior

For details, please see "40.90. Tasked by Superior" on page 906.

41.49. TDL Messaging

For details, please see "40.91. TDL Messaging" on page 907.

41.50. Weapons Pacing/Tracking

Weapons Pacing/Tracking lets you specify that a weapon be marked as pacing, tracking, or neither. Pacing indicates that the status of this item is critical to the mission. Tracking indicates that the commander is interested in the status of this item. These designations do not affect the simulation. They are reported for use by external systems.

➤ **To set pacing or tracking for a unit's weapons:**

1. Select the simulation object.
2. Choose **Set > Disposition > Weapons Pacing/Tracking**. The Weapons Pacing/Tracking dialog box opens.
3. Select the check box for each weapon for which you want to specify pacing or tracking.
4. For each selected weapon type, select an option from the list.
5. Click OK.



VIII. Plans

This section describes how to write plans.

Introduction to Plans

VR-Forces has two kinds of plans: global plans and individual plans. Both kinds of plans can assign tasks and set data requests to a simulation object. Plans can also contain commands that create or delete simulation objects and tactical graphics. The commands in a plan can be conditional; that is, they will be executed only if a condition is satisfied. When you open a scenario, it is in pause mode (unless you start it with the `-l` command line option). When you start the simulation, the plans start executing.

An individual plan is a set of statements that order a particular simulation object to complete a sequence of tasks, change its state, or both. The simulation object carries out the tasks in order unless it is interrupted by an independent task or global command.

A simulation object owns its individual plan. Changing its name, echelon relationship, or force does not change its plan.

Global plans are created and executed independently of any particular simulation object. Commands sent to simulation objects from a global plan affect them as if they received an independent task through the Task menu — they interrupt the current task, if any, and cause the simulation object to abandon its plan, if it has one.

If a simulation object does not have a plan, it does not mean that it has nothing to do. A simulation object may be assigned tasks by its superior in the force hierarchy. It can also be assigned tasks interactively from the GUI. If you want a simulation object to take planned actions independently of those assigned by its superior, you need to add statements to its plan.

You can view a simulation object's plan at any time in the Plan window. The task that a simulation object is currently executing is highlighted. A simple plan might look like:

```
Set Engagement-Rules= "fire-when-fired-upon"
Move to: destination=red point
Wait-Duration:1 min 30 sec
Move to: destination=white point
Set Engagement-Rules="fire-at-will"
```

> ⚠ By default, plans are locked when a simulation is running, but you can choose to make edits by clicking the Unlock button. When you edit a plan during a simulation and then click Apply or OK to apply your changes, however, the simulation object starts executing the plan from the beginning, which may cause its activities to be out of synchronization with the other simulation objects in the simulation.



42. Plan Concepts

This chapter describes plans and plan concepts.

42.1. Introduction to Individual Plans and Global Plans 932
42.2. Conditional Statements 933
- 42.2.1 The If Statement 935
- 42.2.2 Triggers 935
- 42.2.3 While Statements 937
- 42.2.4 Do Until Interrupt Statement 938
- 42.2.5 Wait Until Statements 938
- 42.2.6 Conditional Tests 939

42.3. Plan Variables 943
42.4. Viewing Plans 943
42.5. Considerations for Creating Plans 944
- 42.5.1 Considerations for Using Triggers 944
- 42.5.2 Using Concurrent Tasks in Plans 945
- 42.5.3 Moving In Formation 945
- 42.5.4 Using the Tasked-By-Superior Request in a Plan 945
- 42.5.5 Following Simulation Objects 945
- 42.5.6 Using Non-VR-Forces Simulation Objects in Plans 945

42.1. Introduction to Individual Plans and Global Plans

A plan contains one or more statements that VR-Forces executes without human intervention. You build these statements in dialog boxes. VR-Forces writes the statement to the plan, so you do not have to worry about syntax errors in your statements when you first create them. However, you must understand the meaning of individual commands and parameters and the implications of how you order them, to make sure that a plan does what you want it to do.

VR-Forces supports two types of plans: individual plans and global plans. They have these characteristics:

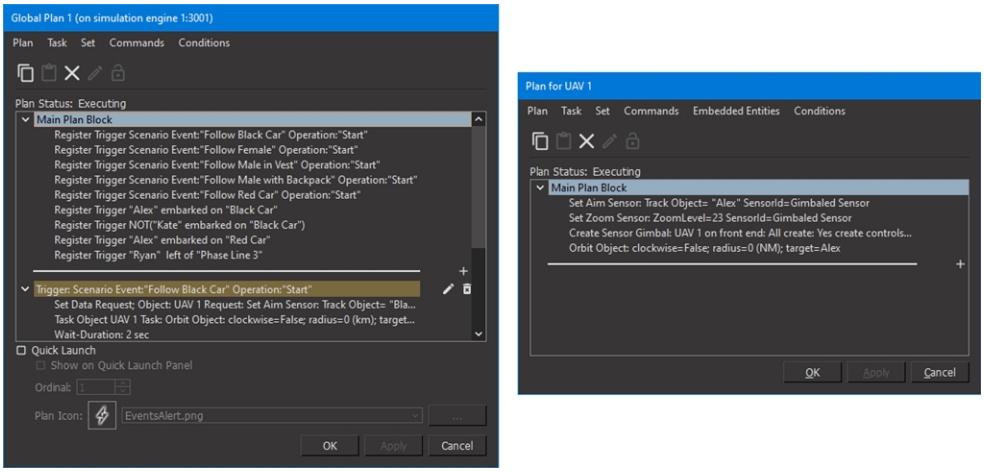
- Individual plans apply to a specific simulation object. They contain tasks and set data requests that are specific to the simulation object and are executed sequentially. The sequence of task execution can be interrupted by receipt of a task from an external actor (a global plan command or independent task). The task sequence can also be interrupted if a trigger (a form of conditional statement) or a reactive task is activated.

  > ⚠ Individual plans can include global commands (which are described later in this section). However, you must be careful not to use global commands, tasks, or sets for a simulation object in its own plan when you really want simulation object-specific tasks or sets. Sending a simulation object a global command, even from within its own plan, terminates its plan.

- Global plans are independent of any simulation object. The statements in a global plan affect the scenario as a whole (create and delete object commands) or a specific simulation object (task and set commands). Tasks sent to a simulation object from a global plan have the same effect on the simulation object as independent tasks assigned by a human player. They interrupt any task that the simulation object is engaged in and terminate its plan. Furthermore, if a global plan has a series of tasks for the same simulation object, they are sent in order without regard to completion of the previous task (as if you issued a series of commands from the Task menu, one immediately after another). In such a case, only the last command sent is likely to be completed.

Figure 301 shows a global plan and an individual plan. Note that the individual plan window (on the right) is tied to a particular simulation object (shown in the title bar).

Figure 301. Global plan and individual plan



42.2. Conditional Statements

Many of the statements used in plans set a parameter or direct a simulation object to perform a specific task unconditionally. However, the plan language also includes conditional statements. A conditional statement specifies a condition and a block of plan statements that are executed if the condition is true. VR-Forces supports these conditional statements:

- **If**. An **If** statement is evaluated once, when the plan execution flow arrives at the statement. If the condition is true at that point, VR-Forces executes the statements in the **If** block. For details, see "42.2.1 The If Statement" on page 935.
- Trigger. A trigger statement lets you plan for a condition without knowing in advance when the condition will occur. It specifies a condition that is evaluated continuously by VR-Forces. If the condition becomes true, VR-Forces executes the statements in the trigger. For details, see "42.2.2 Triggers" on page 935.
- **While**. A **While** statement executes a block of statements repeatedly while a condition remains true. VR-Forces evaluates the condition when the plan first arrives at the **While** statement. If the condition is true, it executes the statements in the block. When it completes the statements, it evaluates the condition again. If it is still true, VR-Forces executes the statements again. It continues to do so as long as the **While** condition is true. For details, see "42.2.3 While Statements" on page 937.
- **Do Until Interrupt**. A **Do Until Interrupt** statement allows you to assign a simulation object to perform a task until a condition takes place. Then the simulation object moves to the next statement in the plan. For details, see "42.2.4 Do Until Interrupt Statement" on page 938.
- **Wait Until**. A **Wait Until** statement causes a simulation object to wait in place until the condition becomes true. Then the simulation object moves to the next statement in the plan. For details, see "42.2.5 Wait Until Statements" on page 938.

The conditions that can be tested by conditional statements are:

- Is a simulation object in a given area? (Entity In Area)
- Is a simulation object within range of its target? (Entity in Range of Target)
- Is a simulation object to the left of a phase line? (Entity Left of Phase Line)
- Has a specific entity been created? (Entity Created)
- Is a simulation object destroyed? (Entity Destroyed)
- Is a simulation object under fire? (Entity Under Fire)
- Has a simulation object located a target? (Entity Has Target)
- Has a simulation object's embarkation state changed? (Entity Embarked)
- Is a simulation object above or below a specified altitude? (Entity Altitude)
- Has this simulation object detected another simulation object? (Detect Entity)
- Has this simulation object received a text message? (Receive Text Message)
- Has an engineering object been breached? (Engineering Object Breached)
- Has this simulation object received a missile approach warning? (Missile Approach Warning)
- Is a scenario event in process or concluded? (Scenario Event)
- Has a lifeform surrendered? (Lifeform Surrendered)
- Does a lifeform have a particular DI-Guy animation? (Entity DI-Guy Animation Check)
- Does a lifeform have a particular DI-Guy appearance? (Entity DI-Guy Appearance Check)
- Is a random probability true? (Random)
- Is a resource for this simulation object at a certain level? (Resource)
- Is a subordinate of a unit independently tasks? (Simulation Object Independently Tasked)
- Is the simulation time greater than or less than a specific elapsed time? (Simulation Time)
- Is the simulation time a certain value? (Simulation Time of Day)
- Is the simulation date and time a certain value? (Simulation Date/Time)
- Is a tactical graphic active? (Tactical Graphic Active)

Conditional statements can test for True and False, and can include the logical operators AND, OR, and NOT.

For details, see "42.2.6 Conditional Tests" on page 939.

42.2.1 The If Statement

An **If** statement tests a condition. It only tests the condition once.

> Do not use an **If** statement to test for receipt of a message or some other circumstance that does not represent a simulation object's state. VR-Forces does not keep a record of events and such a test will almost always evaluate to false. Use a trigger to test for conditions that do not evaluate simulation object state.

An **If** block consists of:

- The word **If**
- A condition to be evaluated
- A block of statements
- An **else** statement
- Optionally, statements for the **else** condition
- An **endif** statement

If the condition is true at the moment the **If** statement is evaluated, control passes to the first statement of the **If** block. If the condition is false, control passes to the first statement in the **else** block, if you have one, or to the first statement after the complete **If** block if you do not have an **else** block. You can nest **If** statements within **If** statements. The syntax is:

```
If (condition) then
  zero_one_or_more_statements
else
  zero_one_or_more_statements
endif
```

When VR-Forces reaches the end of the block of **If** or **else** statements, control passes to the first statement after the **If** statement.

The Add Conditional Expression dialog box ensures that you have the correct syntax. It does not ensure that your **If** block makes logical sense.

42.2.2 Triggers

A trigger consists of a condition and a block of statements that are executed when the condition becomes true.

When you create a trigger, it is not automatically added to the plan's execution sequence. You must register a trigger before VR-Forces will test it. (If you have a programming background, it may be helpful to think of triggers as functions or subroutines that you write separately and then call from elsewhere in a plan. Extending the analogy, a trigger is always local to its plan.)

When you register a trigger, VR-Forces checks the condition. If the condition is true, control passes to the first statement in the trigger body. If the condition is false, VR-Forces continues to the next statement in the plan. It continues to check the trigger condition periodically. If at any point, the condition evaluates as true, control passes to the block of statements associated with the trigger.

When you create a trigger, you can specify that it interrupt the current task or that it suspend and resume the current task.

If you specify that a trigger interrupt the current task, task execution proceeds as follows:

- If a trigger contains any tasks that are mutually exclusive with the currently executing task, the current task is discarded and the statements in the trigger are executed.
- If the tasks in a trigger are not mutually exclusive with the current task, or if the trigger only has set data request statements, the trigger's statements are executed and the current task continues. For more information about this case, see "Using Triggers That Do Not Have Mutually Exclusive Tasks" on page 944.
- After the tasks in the trigger block are completed, control returns to the next statement in the simulation object's plan after the one that was interrupted by the trigger.

If you specify that a trigger suspend the current task, task execution proceeds as follows:

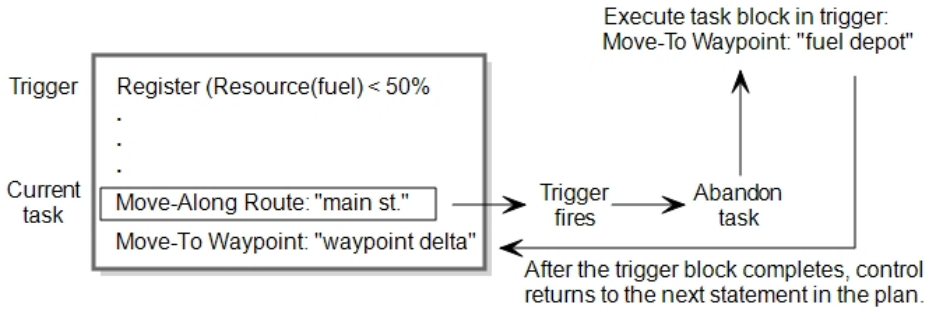
- If a trigger contains any tasks that are mutually exclusive with the currently executing task, the current task is suspended and the statements in the trigger are executed.
- If the tasks in a trigger are not mutually exclusive with the current task, or if the trigger only has set data request statements, the trigger's statements are executed and the current task continues. For more information about this case, see "Using Triggers That Do Not Have Mutually Exclusive Tasks" on page 944.
- After the tasks in the trigger block are completed, the suspended task is resumed, if possible. There may be cases where the previous task is no longer meaningful due to changes in the scenario.

The location of a trigger in a plan has an important effect on its functioning. For a discussion about where to place triggers, see "Deciding Where to Put Triggers" on page 944.)

By default, the name of a trigger is the syntax of its condition. However, you can change the name of a trigger to be any text you want to improve readability or shorten it.

Figure 302 illustrates the process when a trigger has a task that is mutually exclusive with the current task and you have specified that the current task be discarded. After the tasks in the trigger are completed, control returns to the next statement in the simulation object's plan. If the simulation object was in a **While** loop when the trigger fired, it returns to the next statement in the **While** block.

Figure 302. Flow of control for triggers



By default, after a trigger fires, it is removed from the list of registered triggers. If you want to continue to test for the same condition, you must reregister the trigger. You can design the plan in such a way that the trigger is registered again. Alternatively, when you create a trigger, you can specify that it reregister automatically.

> An alternative to using triggers in a plan is to write a reactive task. For information about reactive tasks, see "Chapter 36. Creating Scripted Tasks and Sets" on page 743.

42.2.3 While Statements

A **While** statement consists of:

- The word **While**
- A condition to be evaluated
- A block of statements
- An **endWhile** statement

If the condition is true at the moment the **While** statement is reached, control passes to the first statement of the **While** block. If the condition is false, control passes to the first statement after the complete **While** block. You can nest **While** statements within other conditional blocks.

When VR-Forces reaches the end of the **While** block, it evaluates the condition again. If the condition is still true, VR-Forces executes the statements again. If it is false, control passes to the first statement after the **While** block.

- VR-Forces re-evaluates a **While** condition only after it has executed all of the statements in the block. If the condition becomes false while VR-Forces is executing statements in the block, VR-Forces continues to execute all statements because it does not re-evaluate the condition until it completes the **While** block.
- If you put a continuous task such as Wait or Patrol Between in a **While** block, it is possible that the condition will never be re-evaluated, because these tasks never finish.

42.2.4 Do Until Interrupt Statement

A **Do Until Interrupt** statement allows you to have a simulation object perform a task, even one that otherwise could continue indefinitely. It consists of:

- The words **Do Until Interrupt**
- A condition to be evaluated
- An **End Do Until Interrupt** statement

If the condition is true at the moment the **Do Until Interrupt** statement is reached, the simulation object continues to the next task in the plan. If the condition is false, the simulation object performs the task(s) in the **Do Until Interrupt** statement. VR-Forces continually checks the condition until the condition becomes true or the simulation object is tasked by some other form of input, such as a trigger or independent task. If the condition becomes true, control passes to the next statement in the plan.

If a **Do Until Interrupt** state is interrupted by a trigger, the plan returns to the **Do Until Interrupt** task(s) after the trigger's tasks are completed.

42.2.5 Wait Until Statements

A **Wait Until** statement consists of:

- The words **Wait Until**
- A condition to be evaluated

If the condition is true at the moment the **Wait Until** statement is reached, the simulation object continues to the next task in the plan. If the condition is false, the simulation object waits in place. VR-Forces continues to check the condition periodically until the condition becomes true or the simulation object is tasked by some other form of input, such as a trigger or independent task.

If a **Wait Until** state is interrupted by a trigger and the trigger's Interrupt Current Task check box is not selected, the plan returns to the **Wait Until** state after the trigger's tasks are completed. If the trigger's Interrupt Current Task check box is selected, the plan moves to the statement after the **Wait Until**.

If the condition becomes true, control passes to the next statement in the plan.

42.2.6 Conditional Tests

This section describes each of the conditions that VR-Forces can test. When you set up conditions, keep in mind that:

- If there is no simulation object with the specified name, the result is always false.
- If the specified simulation object is destroyed, the condition might never be satisfied. Your plan should take this possibility into account.

Detect Entity

The Detect Entity condition tests to see if this simulation object has detected another simulation object at a particular identification level. The identification levels are Detected, Classified, Identified, and Full Knowledge. Identification levels are determined by the detection probability configuration files, based on the distance between the detecting simulation object and the target and for how long the simulation object has been detected. For details about the detection probability files, see "76.2. Detection Tables" on page 1516.

This condition is not supported in global plans.

Engineering Object Breached

The Engineering Object Breached condition returns true if the specified engineering object has been breached. (Aggregate-level scenarios only.)

Entity Altitude

Entity Altitude returns true if the simulation object (or member of a unit) is at a higher or lower altitude than the one specified.

Entity Created

Entity Created returns true when the specified simulation object is created in the scenario.

Entity Destroyed

Entity Destroyed returns true if the specified simulation object is destroyed.

- The definition of destruction for a unit is implementation specific. For the entity-level models provided with VR-Forces, units are considered to be destroyed when all members of the unit are destroyed.
- If you have implemented units locally using the VR-Forces Toolkit, they could have a different definition of destroyed.

Entity DI-Guy Animation Check

The Entity Di-Guy Animation Check condition tests the current DI-Guy animation for a lifeform. This condition may be useful if you are scripting a series of animations. For example, if a particular entity is using a threatening or hostile animation, you might want to have other simulation objects respond a certain way.

Entity DI-Guy Appearance Check

The Entity Di-Guy Appearance Check condition tests the current DI-Guy appearance for a lifeform. This condition may be useful if you are scripting a lifeform's interaction with another lifeform based on ethnicity or how they are dressed.

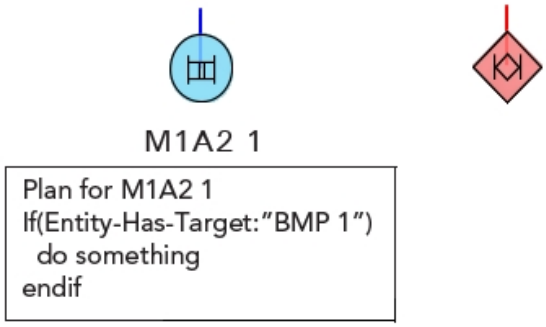
Entity Embarked

Entity Embarked returns true if the named simulation object is embarked.

Entity Has Target

Entity Has Target returns true if the simulation object specified in the condition identifies a target. For example, consider the entities in Figure 303. The condition in M1A2 1's plan is true when BMP 1 has a target, not when M1A2 1 targets BMP 1.

Figure 303. Entity Has Target condition



If a simulation object's rules of engagement prevent it from firing, this condition does not return true when the simulation object identifies a target.

Entity In Area

Entity In Area returns true if the simulation object reports that it is in the named area. You can use the **NOT** operator, to test whether a simulation object is outside the area.

If a simulation object enters an extruded area, VR-Forces does a 2D check first, to see if it is in the area. Then it checks the simulation object's altitude to see if it is within the area's volume.

If you have a plan in which you are testing for entities in an area and destroying them as they enter it, the Exclude Destroyed Simulation Objects option lets you reregister a trigger and only test for entities that have not been destroyed.

Entity In Range of Target

Entity In Range of Target returns true if the simulation object comes within the specified range of the selected simulation object.

Entity Left of Phase Line

Entity Left of Line returns true if the simulation object reports that it is to the left of the specified phase line.

A simulation object is considered to be to the left of a phase line if it is on the left side of an infinite extension of the phase line as viewed from a point on the line facing in the direction of the line. VR-Forces checks the closest segment to the point it is checking to verify whether the entity is to the left of that segment. For a description of phase line direction, see "45.3. Phase Lines" on page 996.

Entity Under Fire

Entity Under Fire returns true if the simulation object is under fire. An entity is considered to be under fire if it is directly targeted in a Fire PDU or if there is a hostile detonation within the **under-fire-parameter** distance specified in the **under-fire-determination-sensor** sensor in its OPE file. The default for all entities is 1000 meters.

 Lifeform Surrendered

The Lifeform Surrendered condition tests to see if a lifeform entity has surrendered (using the Surrendered set data request).

Missile Approach Warning

The Missile Approach Warning condition returns true if a hostile missile is within the specified distance of this simulation object. This condition is not available in global plans.

Random

You can use the Random condition to randomly decide a simulation object's behavior at run time.

`Percentage` – A floating point probability percentage value between 0% (zero) and 100%. Random returns true if a random number between zero and 100 is less than or equal to the specified value.

If you use the Random condition, consider:

- Random is not very useful in triggers, since triggers may evaluate their condition expressions as often as once per tick. This means that a trigger expression of "Random Probability:50%" is likely to fire within the first couple of times it is evaluated. Therefore, we recommend that you not create triggers that are based on a random condition.
- If you use a random condition in a cascading sequence of **If** statements, and you want to ensure the correct probability distribution, be careful how you specify the percentage value. For example, randomly selecting one of three paths with equal probability would be done like this:

```
If (Random Probability:33.33%) then
  Move-Along Route:"1"
Else
  If (Random Probability:50%) then
    Move-Along Route:"2"
  Else
    Move-Along Route:"3"
  Endif
Endif
```

Receive Text Message

The Receive Text Message condition tests to see if this simulation object has received a text message that contains the text specified in the Receive Text Message dialog box. This condition is not available in global plans. The text message can be received as the result of a Send Text Message task or the Send Text Message command in a plan.

Resource

The Resource condition tests the amount of a specified resource for this simulation object using comparison operators. This condition is not available in global plans.

Scenario Event

The Scenario Event condition returns true if the specified scenario event is in process or has completed.

Simulation Object Independently Tasked

The Simulation Object Independently Tasked condition tests to see if an object that is a member of a unit is executing a task independently of the unit. If a unit member is independently tasked, you might not want to issue a new task to the unit until the subordinate finishes its task. This condition lets you test the independent task status of all members of a unit or particular members of a unit to help plan the timing of unit tasks. For example, in the Parachuting example scenario, the plan for the helicopter CH 47 1 waits until all members of Inf SQD 1 are not independently tasked before it sends a message to the squad.

Simulation Time

The Sim Time condition returns true if the elapsed simulation time is greater than (Sim Time >) or less than (Sim Time <) the time specified in the statement. Sim Time is the elapsed simulation time in days, hours, minutes, and seconds.

Simulation Date/Time

The Simulation Date/Time condition returns true if the simulation date and time matches the specified value.

Tactical Graphic Active

The Tactical Graphic Active condition returns true if a published tactical graphic is in the active state. For details about setting a tactical graphic's state, see "47.11. Active and Inactive Tactical Graphics" on page 1035 and "48.2. Activation" on page 1038.

Boolean Operators

You can use these boolean operators in conditional expressions:

- **True**.
- **False**.
- **AND** (*expression, expression*) - evaluates as true if both conditional expressions are true.

- **OR** (*expression, expression*) - evaluates as true if either of the expressions are true.
- **NOT** (*expression*) - evaluates as true if the conditional expression is false.

42.3. Plan Variables

Many tasks require you to specify a simulation object to act on, such as Fire on Target. Conditional tests, such as Entity Destroyed, require you to identify a simulation object to evaluate. However, you will not always know the name of the simulation object that you need to specify. Variables let you specify a simulation object without knowing its name. Plans can reference two variables:

- Simulation Object. The Simulation Object variable references a simulation object that has been identified by a conditional test, such as Entity in Area or Entity Left of Line. For example, suppose that if an opposing force entity enters an area you want to fire at it. You can create a trigger using an Entity in Area condition. Then in the body of the trigger, you can insert a Fire at Target task that specifies the Simulation Object variable rather than the name of a specific entity.

  In an AND condition, the variable is set to the last condition satisfied. In an OR condition, the variable is set to the first condition satisfied.

- Created Object. The Created Object variable references the most recently created object. For example, suppose you create an object from a plan and do not give it a name. You can issue it a plan or refer to it using the Create Object variable. This variable refers to the most recently created object.

Plan variables are specific to each simulation object plan. A variable value in one plan does not affect the variable value in a different plan. Therefore, if you want to use the Created Object variable, you must create the object in the plan in which you want to use it. You could not, for example, create an object in a global plan and then reference it using the Created Object variable from a simulation object's plan.

Once a variable has a value, you can use that variable in a plan as many times as you want to. However, you must keep track of plan logic that might change the value as the simulation object progresses through the plan.

For details about how to use variables in plans, see "43.5. Using Plan Variables" on page 965.

42.4. Viewing Plans

You can view a simulation object's plan and observe its progress through the plan. Statements are numbered to show the order in which they will execute unless interrupted by a trigger, independent task, or other external input. If a plan is executing the statements in a conditional block, VR-Forces draws a box around the block and the numbering is shown for the statements in the block.

If the Plan window is collapsed (that is, the Trigger pane is not visible) and control passes to a trigger, the window automatically expands to show the tasks in the trigger that is being executed.

The task that a simulation object is executing is highlighted in the Plan window. When the simulation object completes its plan, a message is displayed in the status area at the bottom of the Plan window.

You can view the plans for as many simulation objects at a time as you want. In addition to viewing plan status in the Plan window, you can view the current task on the Task Status page of the Information dialog box. The Last Selected Object Panel also lists the currently executing task.

➤ **To view a simulation object's plan:**

1. Select the simulation object whose plan you want to view.
2. Open the Plan window using one of the following methods:
   - Choose **Objects > Plan**.
   - Press **P**.
   - On the Last Selected Object Panel or Command Panel, click the Plan button ().

42.5. Considerations for Creating Plans

This section has tips for constructing plans including how to avoid problems that are hard to debug and how statements affect each other, with implications for how to organize the statements in a plan.

42.5.1 Considerations for Using Triggers

Triggers are a versatile part of plans and you should have a good understanding of their intricacies before you use them.

Deciding Where to Put Triggers

VR-Forces does not evaluate a trigger condition until it registers the trigger. If you want to ensure that a trigger is always evaluated by VR-Forces, register it at the beginning of a plan before any non-trigger statements. If you register a trigger later in a plan, it is possible that it will never be evaluated. This could occur if a plan is abandoned. Of course, you might decide that you do not want a trigger to be evaluated unless a set of preceding tasks have been completed. In that case, placing it later in the plan may be exactly what you want to do.

You can also register a trigger inside a conditional block so that it is registered only if a particular condition is true at a particular time.

For example, if you always want entity 1 to respond to entry of hostile forces into an area, put a trigger at the beginning of the plan. If you want entity 1 to respond to a hostile simulation object in an area only after the simulation has progressed for 15 minutes, nest the trigger inside another trigger that fires when Sim Time > 15 minutes.

Using Triggers That Do Not Have Mutually Exclusive Tasks

A trigger does not need to have any task statements in its statement block. It could have only Set statements, or it could have sets and concurrent tasks. If a trigger block has only Set statements, it does not interrupt the task that the simulation object is executing when

the trigger is executed. The Set statements are executed while the simulation object is performing its task. This has the same effect as if you manually set the simulation object's state while it was performing a task.

Using a trigger that just has Set statements can be useful if you want to change a simulation object's state under certain conditions, but do not know in advance when those conditions might exist.

42.5.2 Using Concurrent Tasks in Plans

If you are assigning tasks to simulation objects manually, you can give a simulation object a concurrent task while it is executing some other task. You cannot do this directly in a plan because you cannot assign two tasks at the same time. A plan always waits for its current task to complete before it starts the next task, even if the next task does not conflict with the current one. If you want to assign a concurrent task in a plan, you must use a trigger. If a trigger that just has a concurrent task fires, the simulation object executes the concurrent task and continues its existing task. If the trigger has other tasks, the behavior will be as expected for triggers.

For more information about concurrent tasks, see "29.1.4 Concurrent Task Execution" on page 561.

As an alternative to using plan triggers to automatically assign concurrent tasks, you can create reactive tasks. For information about reactive tasks, see "29.5. Reactive Tasks" on page 572.

42.5.3 Moving In Formation

If you want a unit to move in a certain formation, you can set the formation (Formation set data request) or you can have the unit move into a formation (Move Into Formation task). The default formation is a column.

42.5.4 Using the Tasked-By-Superior Request in a Plan

If you put the Tasked-By-Superior request in the body of a plan, it has no real effect. This is because, as soon as it is set, the plan goes to the next task and overrides the superior. The only way that the Tasked-By-Superior command is implemented as part of a plan is if it is the last statement in a plan or the last statement in a trigger that happens to execute after all the statements in a simulation object's plan have been executed.

42.5.5 Following Simulation Objects

If you tell two or more simulation objects to follow another simulation object, do not give them the same offset values. If the followed simulation object stops, one follower will arrive at the specified offset position and stop, while the others will circle the location because they cannot all occupy the same space.

42.5.6 Using Non-VR-Forces Simulation Objects in Plans

You can use a non-VR-Forces simulation object as a parameter in a plan, for example, following a remote simulation object, or testing to see if a remote simulation object is in an area.

To identify non-VR-Forces entities, VR-Forces uses their marking text. If a non-VR-Forces simulation object does not have marking text, VR-Forces uses its simulation object identifier as its name.

VR-Forces does not recognize remote units as units, so it cannot check their children. Remote units are ignored by the simulation engine.

- Referring to remote entities by their marking text may have unpredictable results if there is more than one remote simulation object with the same marking text.
- If you include a non-VR-Forces simulation object in a plan you must ensure that the remote simulator uses the same name for the simulation object in future runs.



43. Writing Plans

This chapter explains how to combine tasks, set data requests, global commands, and conditions into plans.

43.1. Writing Plans for Simulation Objects 948
43.1.1 Adding a Task or Set Data Request to a Plan 949
43.1.2 Adding an If Block to a Plan 950
43.1.3 Adding a While Block to a Plan 950
43.1.4 Adding a Wait Until Statement to a Plan 951
43.1.5 Adding a Trigger to a Plan 951
43.1.6 Registering Triggers 953
43.1.7 Cleaning Up Unused Triggers 954
43.1.8 Editing a Statement 955
43.1.9 Deleting Statements from a Plan 955
43.2. Printing Plans 955
43.3. Saving Changes to a Plan 955
43.4. Building Condition Statements 956
43.4.1 Building a Condition Statement with One Condition 956
43.4.2 Building a Complex Condition 957
43.4.3 Editing Condition Statements 962
43.4.4 Specifying Names or Patterns in Condition Statements 963
43.5. Using Plan Variables 965
43.6. Creating Global Plans 966
43.6.1 Controlling When Global Plans Run (Quick Launch) 968
43.6.2 Opening an Information Dialog Box for a Global Plan 969
43.7. Adding Commands to a Plan 969
43.7.1 Adding Tasks and Sets from the Commands Menu 969
43.7.2 Sending Text Messages from the Commands Menu 972
43.7.3 Sending Console Messages 972
43.7.4 Creating Objects from Within a Plan 973

43.7.5 Deleting Objects from Within a Plan 973
43.7.6 Issuing a Plan 973
43.7.7 Sending View Control Messages 975
43.7.8 Creating a Sensor View from a Plan 978
43.7.9 Sending Test Results 979
43.8. Writing Plans for Units 980
43.9. Writing a Plan for Multiple Simulation Objects 980
43.10. Writing Plans for Remote Entities 980
43.11. Copying Plans and Plan Statements 981
43.12. Restarting a Plan 981
43.13. Abandoning a Plan 982
43.14. Editing a Plan During a Simulation 982

43.1. Writing Plans for Simulation Objects

This section explains how to write individual plans. For conceptual information about plans, see "Introduction to Plans" on page 930.

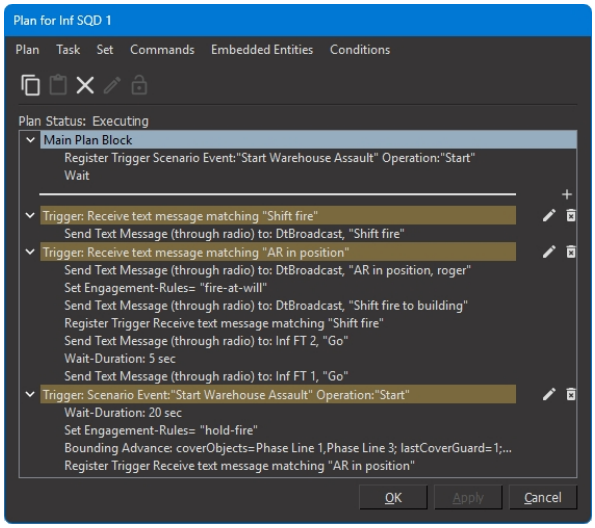
When you write a plan, keep in mind that:

- If you edit a plan during a simulation, when you apply the changes, the simulation object whose plan has been changed begins executing it from the beginning. This might mean its activities are no longer synchronized with those of the other simulation objects, unless the updated plan only includes statements that you want to execute starting at the current point of the simulation. Therefore, we recommend that you edit plans when you first load a scenario, before you start running the simulation.
- If you have two or more Plan windows open at the same time and you then open a dialog box, such as Move To, but do not complete the command, you cannot open that same dialog box from another Plan window until you complete the one that is open.

➤ **To edit a simulation object's plan:**

1. Select a simulation object.
2. Choose **Objects > Plan**, or press **P**. The Plan window opens. The simulation object's plan displays.

Figure 304. Plan window



> You can also open the Plan window by clicking the Plan button ( ) on the Last Selected Object Panel or the Command Panel.

3. Add new statements, edit statements, or delete statements as described in the following sections.

43.1.1 Adding a Task or Set Data Request to a Plan

When you add statements to a plan, you add them after the currently selected statement.

➤ **To add a task to a plan:**

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert a new statement. If the plan does not have any statements yet, the plan root – **Plan** is already selected.
3. Choose **Task >** *category* **>** *Task*, where *category* is one of the categories on the Task menu and *Task* is a task listed for the selected category. A dialog box appropriate to the task opens. The procedures for filling out the task parameters are described in the chapters in "Section VI. Tasks" on page 553.
4. Complete the required parameter information.
5. Click OK in the task dialog box to add the statement. You can click Cancel at any time to exit a task dialog box.

➤ To add a set data request to a plan, follow the same procedure as for adding a task, except choose **Set >** *category* **>** *Set Data Request*.

The procedures for filling out Set statement parameters are in the chapters in "Section VII. Set Data Requests" on page 861.

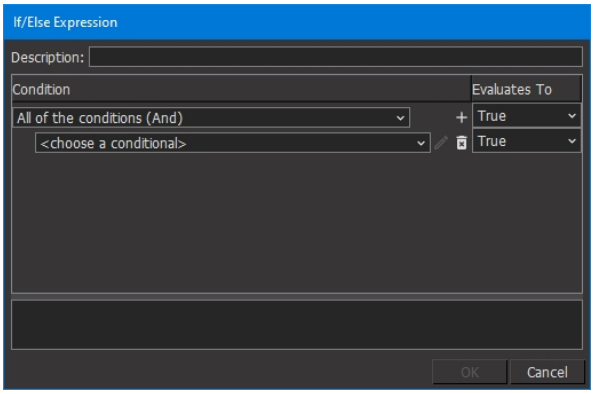
Some task and set statements, such as Wait or Reorganize, do not require you to specify any values. They are added immediately to the plan.

43.1.2 Adding an If Block to a Plan

➤ **To add an If condition to a plan:**

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the **If** block. If the plan does not have any statements yet, the plan root – **Plan** is already selected.
3. Choose **Conditions > If**. The If/Else Expression dialog box opens (Figure 305).

Figure 305. If/Else Expression dialog box



4. Optionally, in the Description box, type a description to use in the If statement instead of the condition syntax.
5. Create the conditional expression as described in "43.4. Building Condition Statements" on page 956.
6. Click OK. An **If** statement block is added to the plan.
7. Select the first line in the block (the **If** statement).
8. Add the tasks, sets, or other commands that you want to be executed if the condition is true.
9. Optionally, select the **else** line and add the commands you want to be executed if the condition is false.

43.1.3 Adding a While Block to a Plan

➤ **To add a While block to a plan:**

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the **While** block. If the plan does not have any statements yet, the plan root – **Plan** is already selected.

3. Choose **Conditions > While**. The While Expression dialog box opens. (It is identical to the If/Else Expression dialog box (Figure 305).)
4. Optionally, in the Description box, type a description to use in the While statement instead of the condition syntax.
5. Create the conditional expression as described in "43.4. Building Condition Statements" on page 956.
6. Click OK. A **While** statement block is added to the plan.
7. Select the first line in the block (the **While** statement).
8. Add the tasks, sets, or other commands that you want to be executed if the condition is true.

43.1.4 Adding a Wait Until Statement to a Plan

➤ **To add a Wait Until statement to a plan:**

1. Open the Plan window for a simulation object.
2. In the Plan window, select the statement immediately above where you want to insert the **Wait Until** statement. If the plan does not have any statements yet, the plan root – **Plan** is already selected.
3. Choose **Conditions > Wait Until**. The Wait Until Expression dialog box opens. (It is identical to the If/Else Expression dialog box (Figure 305).)
4. Optionally, in the Description box, type a description to use in the **Wait Until** statement instead of the condition syntax.
5. Create the conditional expression as described in "43.4. Building Condition Statements" on page 956.
6. Click OK. A **Wait Until** statement is added to the plan.

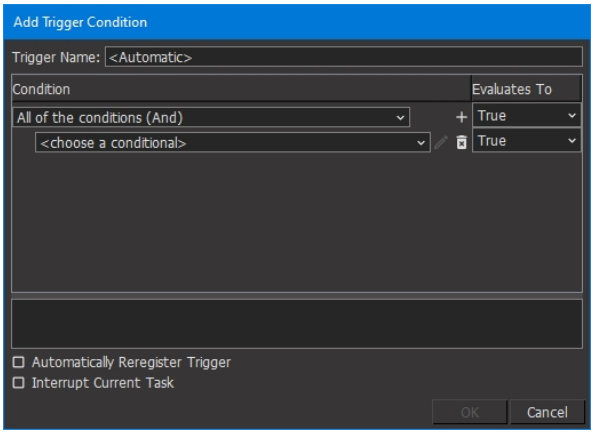
43.1.5 Adding a Trigger to a Plan

Adding a trigger to a plan is a three-step process. First, you specify the trigger condition. Then you add the commands to be executed to the trigger body. Finally, you register the trigger in the body of the plan.

➤ **To add a Trigger to a plan:**

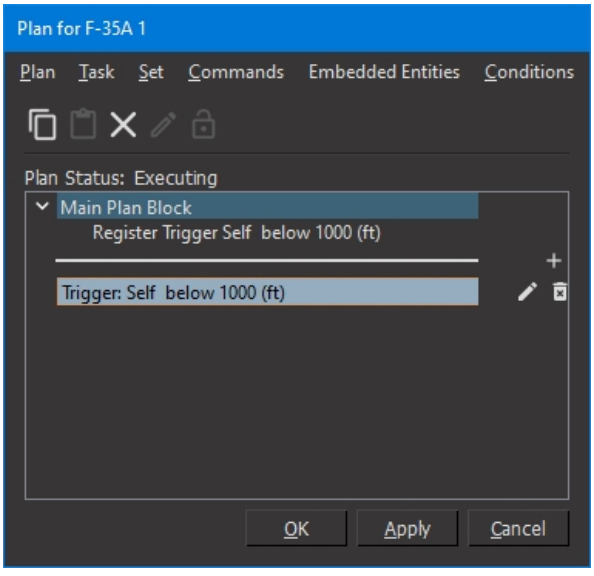
1. Open the Plan window for a simulation object.
2. Choose **Conditions > Add Trigger**. The Edit Trigger Condition dialog box opens (Figure 306). (You can also add a trigger by choosing **Conditions > Register Trigger** and selecting New in the trigger list.)

Figure 306. Trigger dialog box



3. Optionally, in the Trigger Name box, enter a name for the trigger. If you specify a name, it displays in the Register command when you register the trigger. If you do not specify a name, the text of the condition displays in the Register command.
4. Optionally, select the Automatically Reregister Trigger check box to specify that the trigger should be automatically reregistered after it runs.
5. Optionally, select the Interrupt Current Task check box to specify that the trigger should interrupt the currently running task rather than suspending and resuming it.
6. Create the conditional expression as described in "43.4. Building Condition Statements" on page 956.
7. Click OK. The trigger name is listed below the main plan block. (Figure 307).

Figure 307. Trigger pane



8. Select the first line in the Trigger pane (**Trigger Body**).

9. Add the tasks, sets, or other commands that you want to be executed when the condition becomes true.
10. Register the trigger. For details, see "43.1.6 Registering Triggers" below.

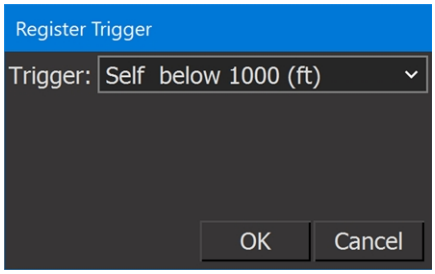
43.1.6 Registering Triggers

Writing a trigger does not necessarily include it in a plan. To use a trigger, you must register it. If you create a new trigger using **Conditions > Add Trigger**, you must register it manually. If you create a new trigger by choosing **Conditions > Register Trigger** and selecting New, it is automatically registered at the currently selected location in the plan.

➤ **To register a trigger:**

1. In the Plan window, select the statement below which you want to register the trigger.
2. Choose **Conditions > Register Trigger**. The Register Trigger dialog box opens (Figure 308).

Figure 308. Register Trigger dialog box



3. Select the trigger you want to register from the list. (If there are no triggers in the plan, or you want to create a new one, select New.)
4. Click OK.

Reregistering Triggers

By default, once the statements in a trigger are executed, the trigger is unregistered and VR-Forces no longer tests the condition. However there may be cases where you want to test for a recurring condition. You could register the trigger again when the trigger block competes. Or, when you create or edit a trigger, you can specify that it automatically reregister. For details, see "43.1.5 Adding a Trigger to a Plan" on page 951.

As an alternative to using triggers, you can write reactive tasks, which work similarly to triggers, or you can write a regular scripted task that is designed to repeatedly test a condition.

Unregistering Triggers

You can unregister triggers from within a plan. Unregistering a trigger does not remove it from a plan. It causes VR-Forces to stop testing the condition. You might want to unregister a trigger in response to a change in scenario circumstances that makes the actions in the trigger undesirable.

➤ **To unregister a trigger:**

1. Select the plan statement above where you want to add an Unregister Trigger statement.
2. In the Plan window, choose **Conditions > Unregister Trigger**. The Unregister Trigger dialog box opens.
3. Select the trigger that you want to unregister.
4. Click OK.

43.1.7 Cleaning Up Unused Triggers

If you add triggers to a plan and then decide not to use (register) them, you can quickly remove them from the plan. (However, there is no penalty for leaving unused triggers in a plan.)

➤ To clean up unused triggers, choose **Plan > Clean Up Unused Triggers**.

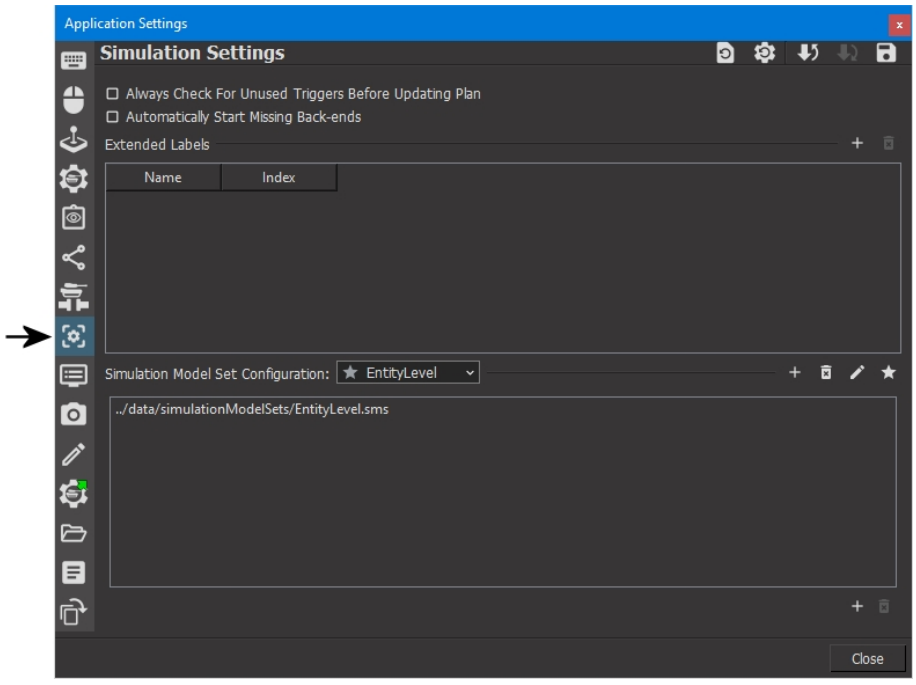
Automatically Checking for Unused Triggers

VR-Forces can automatically check for unused triggers when you save or apply changes to a plan. If it finds any, it displays a dialog box offering you options for handling them.

➤ **To specify that VR-Forces should check for unused triggers when it updates a plan:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Simulation Settings page (Figure 309).

Figure 309. Simulation Settings page



3. Select the Always Check For Unused Triggers Before Updating Plan check box.
4. Click Close.

43.1.8 Editing a Statement

You can edit simple statements in the Plan window.

➤ **To edit a plan statement:**

1. Double-click the statement or select the statement and then click the Edit button (). An edit dialog box for the particular statement type opens.
2. Enter the changes you want to make.
3. Click OK in the edit dialog box.
4. Click OK or Apply in the Plan window.

43.1.9 Deleting Statements from a Plan

You can delete statements from a plan individually, or all at once. You can delete the individual statements in the statement block of a conditional statement. If you delete the conditional statement itself, all substatements are also deleted.

➤ **To delete one statement from a plan:**

1. Select the statement you want to delete.
2. Click the Delete button () or press **Delete**.

➤ **To delete all statements in a plan:**

1. Select the top line in the plan.
2. Click Delete () or press **Delete**.

43.2. Printing Plans

You can print plans. The printout lists the name of the scenario, the simulation object's name, and the plan statements.

➤ **To print a plan:**

1. Open a global plan or the Plan window for the simulation object whose plan you want to print.
2. Choose **Plan > Print**. A standard print dialog box for your operating system opens.
3. Complete the dialog box as you normally would.

43.3. Saving Changes to a Plan

You can save changes to a plan and continue editing, or save changes and exit the Plan window. The plan is saved to the scenario plan file. You do not specify a file to save it to.

➤ To save changes to a plan, in the Plan window, click Apply or OK.

43.4. Building Condition Statements

If you want to add an **If** block, a **While** loop, a **Wait Until**, or a trigger, you must specify a condition to be tested. The conditional expression dialog box is a dynamic dialog box that lets you build a condition. It contains lists that have the permissible options for each step of a condition. As you select options, the dialog box reloads, adding additional fields. You can build condition statements using comparison operators, logical operators, and tests of simulation object status.

A condition statement can have one condition or many. They can be nested and grouped. In the following sections, we explain how to build a simple condition and how to build a complex condition.

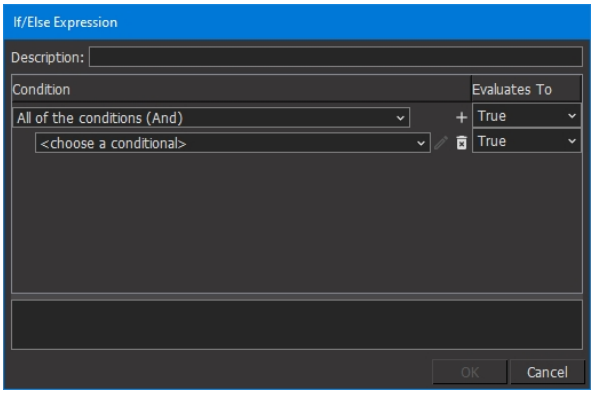
43.4.1 Building a Condition Statement with One Condition

Although the If/Else Expression, While Expression, Wait Until Expression, and Trigger dialog boxes differ slightly, the process of building a condition is the same for all of them.

➤ **To build a condition statement that has one condition:**

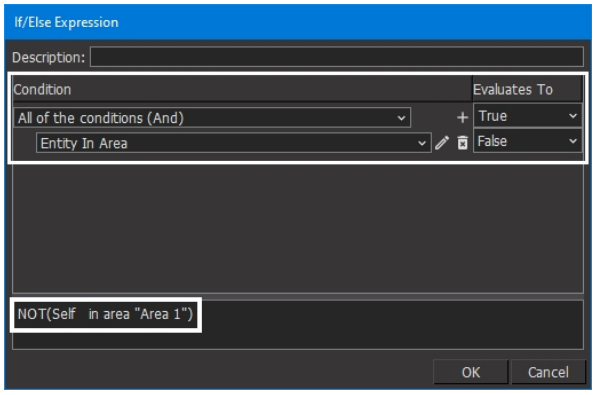
1. On the Conditions menu, choose the type of condition you want to create. The applicable dialog box opens (Figure 310). The top line in the condition specifies that all of the conditions are True. Since we are creating a condition statement that just has one condition, this line is largely irrelevant except for the Evaluates To column.

Figure 310. If/Else Expression dialog box



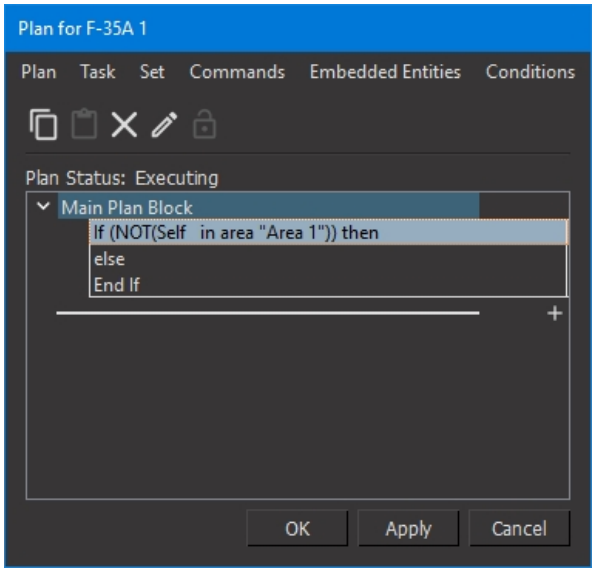
2. Optionally, in the Description box, type a short summary of the condition.
3. Optionally, in the Evaluates To list, change the entire condition from True to False.
4. In the <choose a conditional> list, select the condition that you want to test. If applicable, a dialog box opens for you to specify the parameters of the condition.
5. Specify the parameters of the condition.
6. Click OK. The condition is displayed in the display box at the bottom of the dialog box (Figure 311).
7. Optionally, in the Evaluates To list, change the condition from True to False. Figure 311 shows a simple condition.

Figure 311. Simple conditional expression



8. Click OK. The condition statement is added to the plan or the Trigger pane. In Figure 312, note that the If statement uses the text from the description (in Figure 311) instead of the full syntax of the condition shown at the bottom of the If/Else Expression dialog box.

Figure 312. Simple conditional expression in a plan



43.4.2 Building a Complex Condition

You can build condition statements with multiple conditions, nested conditions, and logical operators. The condition expression dialog box displays the syntax of the condition as you build it. The layout of the window shows which conditions are grouped and which are subordinate to others. For an example of building a complex condition, see "Complex Condition Tutorial " on the next page.

➤ **To build a complex condition:**

1. On the Conditions menu, choose the type of condition you want to create. The applicable condition expression dialog box opens. The top line in the condition specifies that all of the conditions are True.
2. Optionally change the top level logical operator to Or.
3. Optionally change the top level evaluator to False.
4. In the <choose a conditional> list, select the condition that you want to test. If applicable, a dialog box opens for you to specify the parameters of the condition.
5. Specify the parameters of the condition.
6. Click OK. The condition displays in the display box at the bottom of the dialog box.
7. Click the Add button (+) next to the top line of the condition. A new condition line is added.
8. Select a condition to test and specify its parameters.
9. Continue to add conditions as desired. The condition syntax updates as you add conditions.
10. Optionally, select a new logical operator in the conditions list (And or Or). A subordinate condition line is added under the logical operator and indented.
11. Add conditions for the logical operator group as you did for the top level logical operator. To simplify the view, you can expand and collapse the logical grouping by clicking the arrow to the left of the logical operator line.
12. Continue adding conditions as desired.
13. Click OK.

Complex Condition Tutorial

This section walks you through the process of creating a complex condition. It creates the following conditional expression:

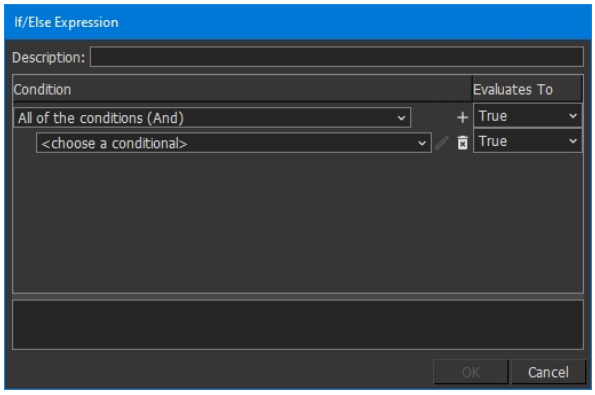
```
  If("M1A2 4" in area "Area 1" AND ("M1A2 2" left of "Phase Line 1" OR
Entity "T80 1" destroyed))then
  else
  endif
```

The procedure assumes a scenario with four M1A2 entities, a T-80, a phase line, and an area.

➤ **To build the condition statement:**

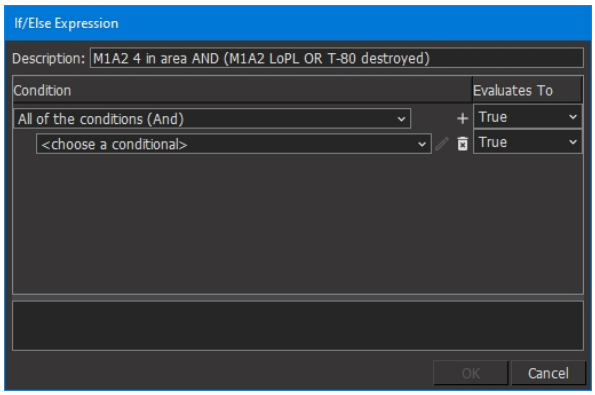
1. Open the Plan window for M1A2 1.
2. Choose **Conditions > If**. The If/Else Expression dialog box opens (Figure 313).

Figure 313. If/Else Expression dialog box



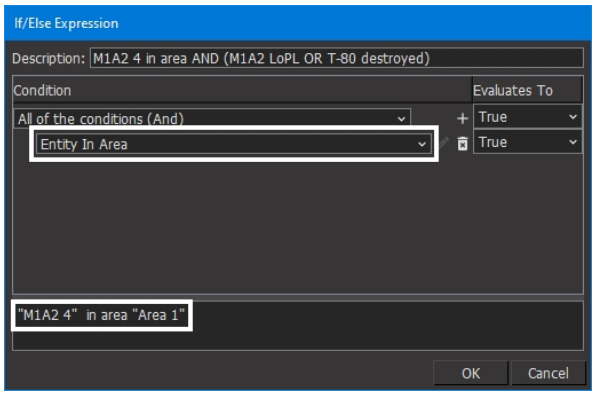
3. Type a description for the expression you want to build (Figure 314).

Figure 314. Description added



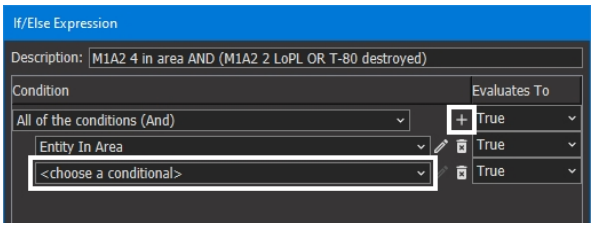
4. In the <choose a conditional> list, select **Entity in Area**. The Entity in Area dialog box opens.
5. In the Entity in Area dialog box, select M1A2 4 and Area 1.
6. Click OK. The condition is added and the If expression is updated (Figure 315).

Figure 315. Entity in Area condition added



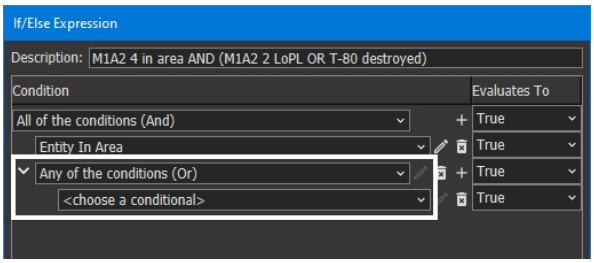
7. On the All of the conditions line, click the Add button (+). A new line is added to the window (Figure 316).

Figure 316. Condition added



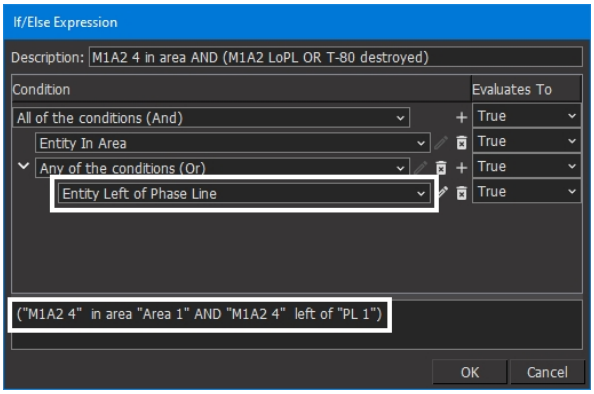
8. In the <choose a conditional> list, select **Any of the conditions (Or)**. A new line is added to the window (Figure 317).

Figure 317. Or operator and condition added



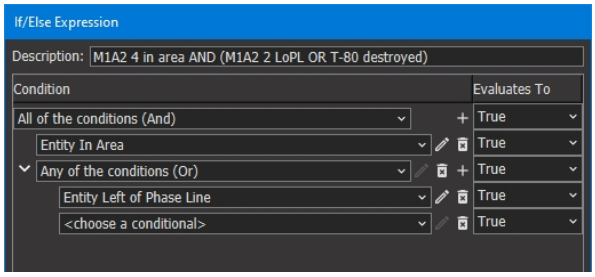
9. In the <choose a conditional> list, select **Entity Left of Phase Line**. The Entity Left of Phase Line dialog box opens.
10. Select M1A2 2 and Phase Line 1.
11. Click OK. The condition is added and the If expression is updated (Figure 318).

Figure 318. Entity Left of Phase Line condition specified



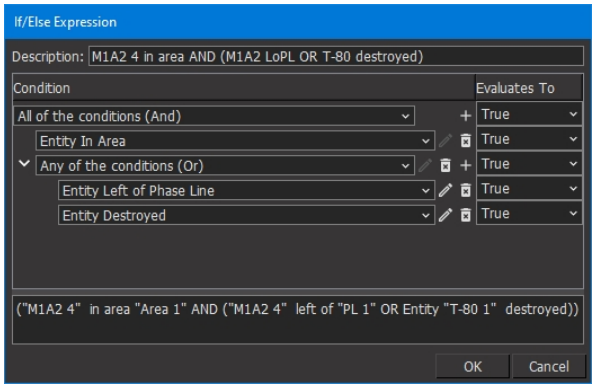
12. On the Any of the conditions (Or) line, click the Add button. A new line is added to the window (Figure 319).

Figure 319. Last condition line added



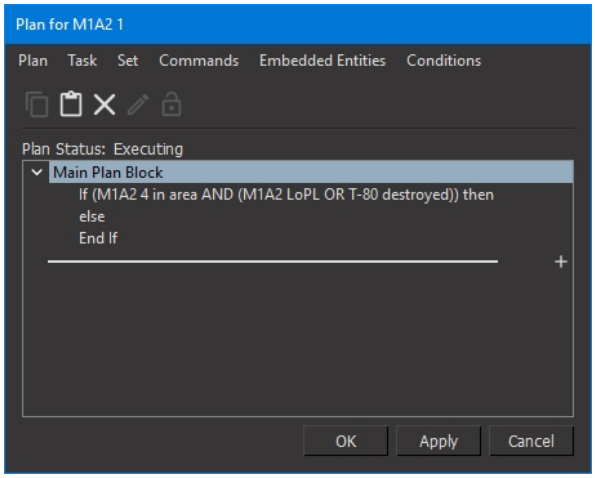
13. In the <choose a conditional> list, select **Entity Destroyed**. The Entity Destroyed dialog box opens.
14. Select T-80 1.
15. Click OK. The condition is added and the If expression is updated (Figure 320).

Figure 320. Completed expression



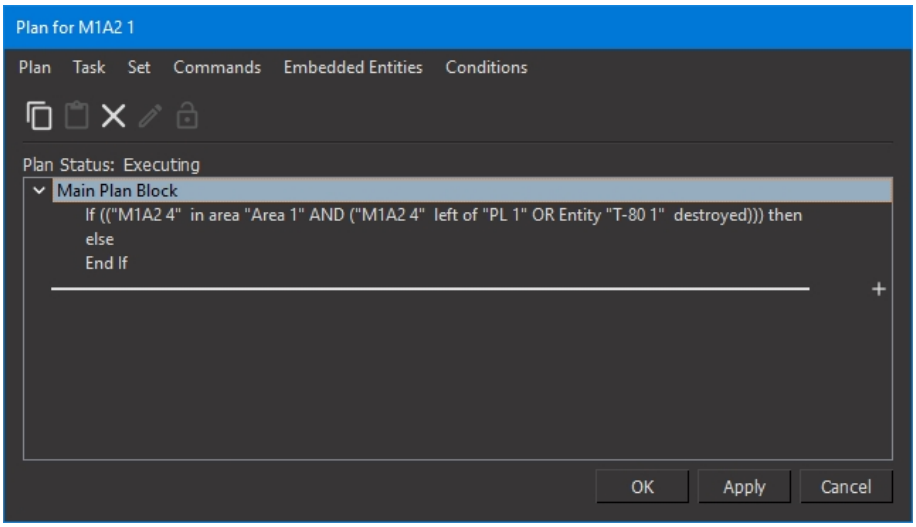
16. In the If/Else Expression dialog box, click OK. The If expression is added to the plan window. Notice that the expression uses the text from the Description box (Figure 321).

Figure 321. If expression using condition description



17. If you did not include a description, the entire syntax of the expression displays (Figure 322).

Figure 322. If expression using expression syntax



43.4.3 Editing Condition Statements

➤ **To edit an If, While, or Wait Until condition statement:**

1. In the Plan window, double-click the statement. The applicable expression building dialog box opens.
2. To edit the parameters of a condition, click the Edit button (✎) for that condition and change the parameters as desired.
3. To change the condition being tested, select a different condition in the list and specify its parameters.
4. To delete a condition, click the Delete button (🗑) for that condition.
5. To change how the condition is evaluated, choose a different option in the Evaluates To list.
6. Click OK.

➤ **To edit a Trigger:**

1. In the Triggers list, found below the Main Plan Block, select the trigger that you want to edit.
2. Click the Edit button (✎). The Trigger dialog box opens.
3. To edit the parameters of a condition, click the Edit button (✎) for that condition and change the parameters as desired.
4. To change the condition being tested, select a different condition in the list and specify its parameters.
5. To delete a condition, click the Delete button (🗑) for that condition.

6. To change how the condition is evaluated, choose a different option in the Evaluates To list.
7. Click OK.

43.4.4 Specifying Names or Patterns in Condition Statements

Many of the conditions that you can use to build a condition statement require you to specify simulation objects for which the condition will be tested. You can specify:

- A specific named simulation object, for example, simulation object M1A2 1 is in Area 1.
- That the condition applies to "Self", for example, "I am in Area 1". Specifying a condition for one's self makes it easy to copy plans or plan statements and paste them into the plans of other simulation objects without doing any editing.
- An enumeration, for example, any simulation object with enumeration 1:1:225:1:1:3:0 (any M1A2).
- The inverse of the selection, for example, any simulation object but myself, or any simulation object except an M1A2.
- That for units, the entire unit must satisfy the condition (the default), or that any subordinate member of a unit satisfies the condition. For example, the default behavior when Entity Is Destroyed is applied to a unit is that all members of the unit must be destroyed for the condition to be true. The alternative is that the condition is true if any member of the unit is destroyed.

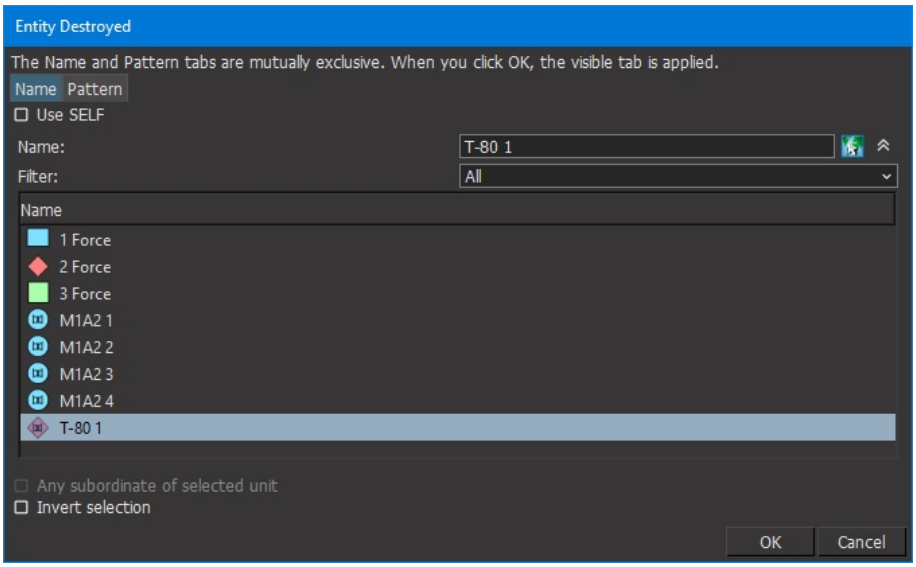
The Name and Pattern specifications are mutually exclusive. You cannot specify a name and a pattern. The tab in the Condition dialog box that is visible when you click OK is the specification that is applied.

Specifying a Name in a Condition Statement

➤ **To specify a named simulation object in a condition statement:**

1. In a condition dialog box, select the Name tab (Figure 323).

Figure 323. Name tab in condition dialog box



2. To specify the simulation object whose plan this is as the simulation object in the condition, select the Use SELF check box.

   To specify a simulation object by name:

   a. Optionally filter the list. If you want to use a plan variable, you must select the Plan Variables filter.

   b. Type the name of the simulation object or variable or expand the list and then select it.

3. If you select a unit in the list, the Any Subordinate of Selected Unit check box is enabled. If you want the condition to be satisfied when any member of the unit meets the condition, select the check box.

4. To specify the inverse of the name, that is, any member except the selected one, select the Invert Selection check box.

   > Selecting both Any Subordinate of Selected Unit and Invert Selection causes the condition to be true if it is true for any entity in the scenario.

5. Click OK.

Specifying a Pattern in a Condition Statement

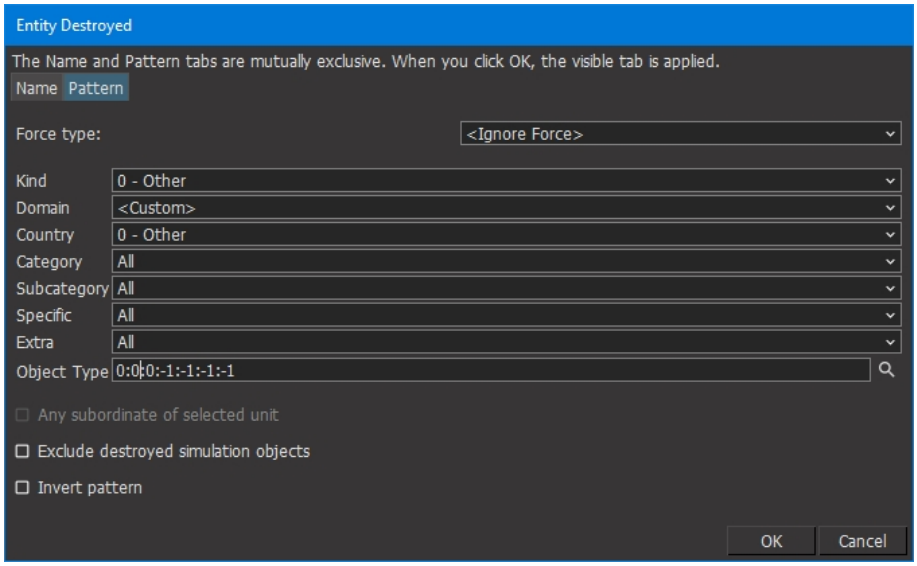
The enumerations used in the lists on the Pattern tab are loaded from the file *./appData/settings/vrfGui/condition-entity-types.csv*. They represent a subset of the enumerations typically used for simulation objects in simulations. However, you can enter any numeric value you want in the enumeration value box.

If you expect that you will consistently need to specify enumeration values that are not part of *condition-entity-types.csv*, such as a particular country code, you can edit it to add the patterns you need.

➤ **To specify a pattern in a condition statement:**

1. In a conditional statement dialog box, select the Pattern tab (Figure 324).

Figure 324. Pattern tab in condition dialog box



2. Optionally, specify a force to test for.
3. For each element of the enumeration, select an option from the list, or type an enumeration in the text box.
4. If the enumeration you enter is for a unit, the Any Subordinate Of Selected Unit check box is enabled. If you want the condition to be satisfied when any member of the unit meets the condition, select the check box.
5. To specify the inverse of the name, that is, any member except the selected one, select the Invert Pattern check box.
6. Click OK.

43.5. Using Plan Variables

You can use plan variables in most places where you would select a specific simulation object in a task, set, or conditional test. Plan variables use the names Simulation Object and Created Object. You cannot add your own variables or change the names.

The variables are assigned values only after an object has been created from within a plan (Created Object) or a conditional test has been met that identifies a simulation object (Simulation Object). If you use a variable in a plan that does not have a value, it is ignored.

A Simulation Object variable is not assigned a value until a condition has been completely evaluated. Therefore, you cannot do something like If Entity In Area AND Simulation

Object is Destroyed and expect that the Simulation Object variable refers to the entity identified by Entity in Area. In this case, the Simulation Object variable would refer to a previously identified simulation object or it would just be empty.

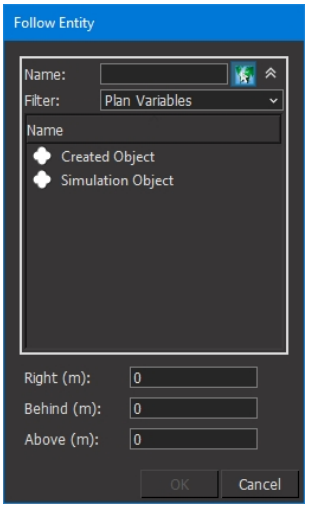
> Remember that the values assigned to plan variables can change as a simulation object progresses through its plan.

For conceptual information about plan variables, see "42.3. Plan Variables" on page 943.

➤ **To use a plan variable in plan statement:**

1. In the statement creation dialog box, select Plan Variables in the Filter list. The dialog box lists Created Object and Simulation Object (Figure 325).

Figure 325. Plan variables



2. Select the variable that you want to use.
3. Complete the rest of the dialog box normally.

43.6. Creating Global Plans

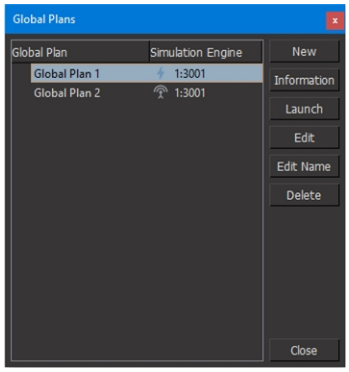
Global plans are not tied to a specific simulation object. However, since they must be executed by a simulation engine, they are assigned to a specific sim engine.

Global plans have a limited set of Task and Set commands and the same set of Commands menu options and conditions as individual plans. (You can add additional tasks using the VR-Forces API or by writing scripted tasks.) You can open an Information dialog box for a global plan to see its task status and console messages.

➤ **To create a global plan:**

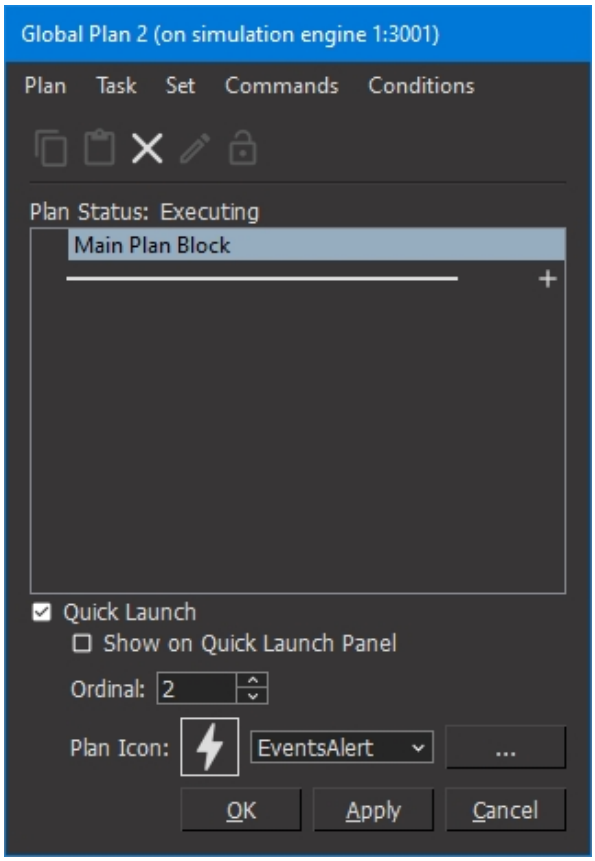
1. Choose **Simulation > Global Plans** or click the Global Plans button ( ) in the Scenario Components Toolbar. The Global Plans window opens (Figure 326). It lists each global plan and the simulation engine on which it is executed.

Figure 326. Global Plans window



2. Click New. The New Global Plan Information dialog box opens.
3. In the Plan Name box, enter a name for the plan.
4. If your scenario has more than one sim engine, select the sim engine on which you want to run the global plan.
5. Click OK. The Global Plan window opens (Figure 327).

Figure 327. Edit Global Plan window



6. Add commands from the Task, Set, or Conditions menu, as described in "43.1. Writing Plans for Simulation Objects" on page 948. Add commands from the Commands menu as described in "43.7. Adding Commands to a Plan" on the facing page.
7. Optionally, select the Quick Launch check box to enable quick launching of this plan. For information about using the Quick Launch Panel, see "43.6.1 Controlling When Global Plans Run (Quick Launch)" below.
8. Optionally, select the Show on Quick Launch Panel check box to add this global plan to the Quick Launch Panel.
9. Optionally, in the Ordinal box, specify the position of this plan on the Quick Launch Panel.
10. Optionally, change the icon for this plan on the Quick Launch Panel:
    a. Click the Browse button next to the icon name. A file chooser window opens.
    b. Select the file you want to use for the icon.
    c. Click Open.
11. Click OK.

43.6.1 Controlling When Global Plans Run (Quick Launch)

If you enable quick launch for a global plan, it does not start executing until you launch it. You can launch it from the Quick Launch Panel, from the Global Plans window, or from another plan. (You can enable quick launch without putting an icon on the Quick Launch Panel.) When you launch a global plan, it runs to the end. When it finishes, you can launch it again.

> The Quick Launch Panel shows an icon and text for each quick launch item. The text is the name of the plan.

➤ To launch a global plan from the Quick Launch Panel, click the icon and text.

➤ **To launch a global plan from the Global Plans window:**

1. Choose **Simulation > Global Plans** or click the Global Plans button ( ) in the Scenario Components Toolbar. The Global Plans window opens (Figure 326).
2. Select the global plan you want to launch. If it is not running, the Launch button becomes active.
3. Click Launch. The global plan runs. When it completes, the Launch button becomes active again.

➤ **To launch a quick launch global plan from a plan:**

1. Open the plan window for an individual plan or a global plan.
2. Choose **Commands > Launch Global Plan**. The Launch Global Plan window opens.
3. Select the quick launch global plan you want to launch.
4. Click OK.

43.6.2 Opening an Information Dialog Box for a Global Plan

➤ **To open an Information dialog box for a global plan:**

1. Choose **Simulation > Global Plans** or click the Global Plans button () in the Scenario Components Toolbar. The Global Plans window opens (Figure 326).
2. Select the global plan for which you want to open an Information dialog box.
3. Click Information.

43.7. Adding Commands to a Plan

The Commands menu lets you assign tasks and set data requests to simulation objects and create the objects on the Create Object Palette. There are also several commands that are only available on the Commands menu. Except as noted in this section, the procedures for adding global commands and conditional statements to a global plan are the same as the procedures for adding them to individual plans.

Since a global plan has no idea what type of simulation object you might want to give a task or set data request to, when you add a task or set, the full list of possible tasks and sets is available. There is no context sensitivity like there is when you select a simulation object and view the Task or Set menu. It is up to you to be sure that the task or set you are assigning to a simulation object is supported by that object.

The best way to determine this is to create a simulation object of the type to which you want to give a task and see if the task or set data request is available on its context-sensitive Task or Set menu. You can also check the Simulation Object Editor to see if a simulation object has the system used for a particular task.

43.7.1 Adding Tasks and Sets from the Commands Menu

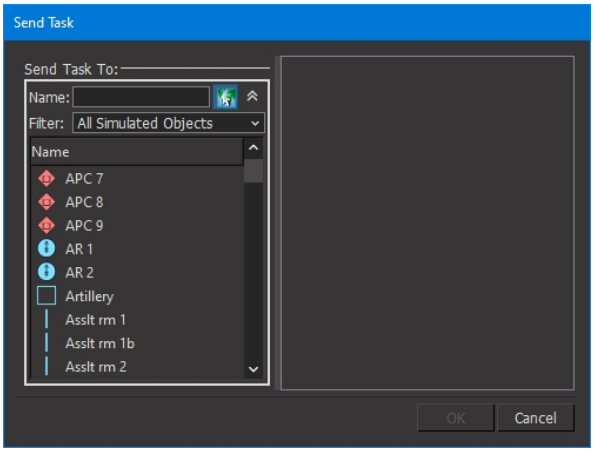
The commands on the Task and Set menus in an individual plan apply implicitly to the simulation object for whom you are writing a plan. Tasks and sets on the Commands menu must be assigned explicitly to a target simulation object.

Other than the requirement to specify the target simulation object, adding a global task or set command to a plan uses the same procedure as adding any task or set to an individual plan.

➤ **To assign a task from the Commands menu:**

1. In a Plan window or Global Plan window, choose **Commands > Send Task**. The Send Task dialog box opens (Figure 328).

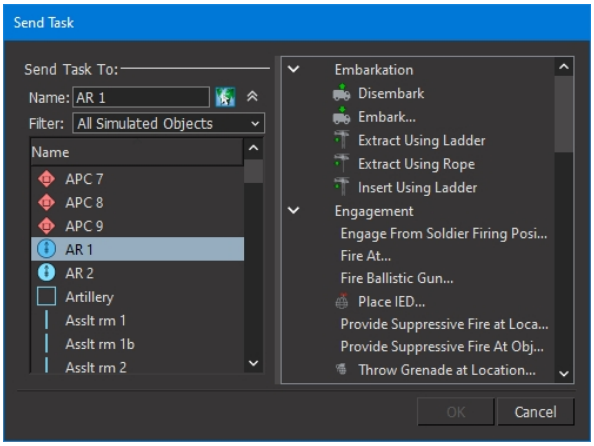
Figure 328. Task dialog box



The procedure for adding a set data request is the same as for a task, except that you choose **Commands > Send Set** and the Send Set dialog box opens.

2. In the Apply Command To group box, select a simulation object from the list or type its name in the Name box. The simulation object does not have to exist in the scenario at the time you add the command, but you should be sure that it will be created before the command is run. The window displays a list of tasks for the selected simulation object (Figure 329). If the simulation object already exists in the scenario, the list of tasks is context-sensitive.

Figure 329. Task list



Remember that a command task affects a simulation object like an independent task. It interrupts the current task and causes the simulation object to abandon its plan. If you are adding a command to an individual plan, be sure that you are assigning it to some other simulation object and not the simulation object for whom you are writing the plan. (Unless you want the simulation object to abandon its plan.)

3. Select the task you want to assign.
4. Click OK. If the task takes parameters, a dialog box opens.

   If the task does not take parameters, such as Wait, the command is added to the plan.
5. If the task takes parameters, specify the parameters and click OK in the task-specific dialog box. See the chapters in "Section VI. Tasks" on page 553 and "Section VII. Set Data Requests" on page 861 for the details of assigning specific tasks and sets.
6. Click OK. The command is added to the plan.

Editing Task and Set Commands in Plans

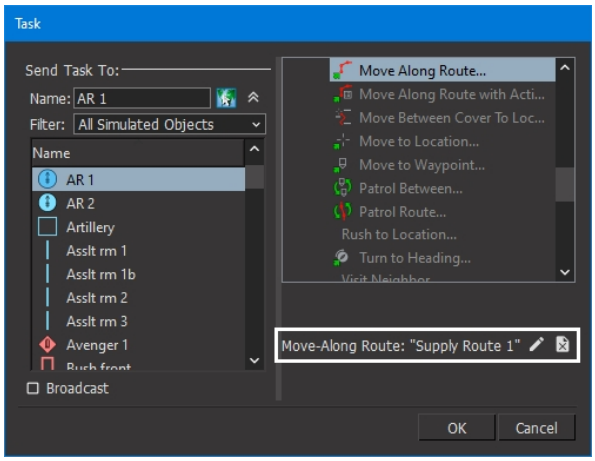
If you want to edit a task or set command in a plan, you can:

- Delete the plan statement and start over.
- Edit the existing statement. If you edit the existing statement, you can:
  - Delete the current task and choose a different one.
  - Edit the parameters of the task.

➤ **To edit a task or set command in a plan:**

1. Select the plan statement you want to edit.
2. Click the Edit button, or double-click the statement. The Send Task or Send Set dialog box opens. The task list is grayed out. If you are not sure of the task that you are editing, you can read the text of the command below the window (Figure 330).

Figure 330. Send Task dialog box for plan command



3. To keep the same task and change the parameters, click the Edit button ( ). The appropriate dialog box opens and you can edit the parameters.

   To clear the task and start over, click the Delete button ( ).
4. Click OK.

43.7.2 Sending Text Messages from the Commands Menu

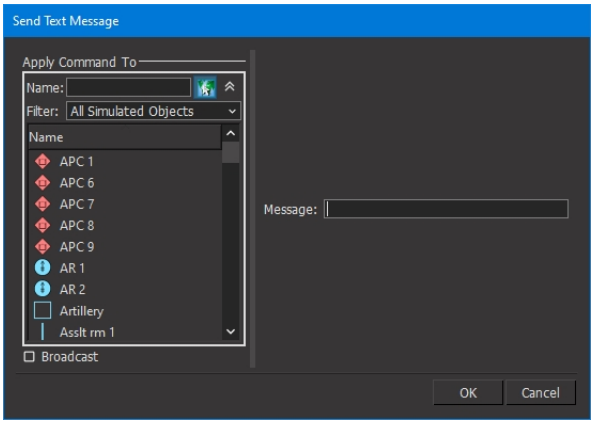
The Send Text Message command lets you send a text message to a simulation object without using the radio network. The receiving simulation object can test for receipt of the message in a conditional statement in its plan. (For details, see "Receive Text Message" on page 942.)

Because the Send Text Message command does not use the radio network, simulation objects in forces other than the sending simulation object can receive the text message. Additionally, text messages sent using the Send Text Message command do not display communication lines and the sending object does not display squawk indicators.

➤ **To send a text message from the Commands menu:**

1. In a Plan window or Global Plan window, choose **Commands > Send Text Message**. The Send Text Message dialog box opens (Figure 331).

Figure 331. Send Text Message dialog box for plan command



2. To send the message to all simulation objects in the scenario, select the Broadcast check box.

   To send the message to a specific simulation object, in the Name text box, type the name of the simulation object that will receive the message, or select the simulation object in the list or on the terrain.

3. In the Message box, type the text message.
4. Click OK. The command is added to the plan.

43.7.3 Sending Console Messages

You can send messages to be displayed in the console on an object's Information dialog box and to the sim engine console. Messages sent by global plans go to the sim engine console. Messages sent from a simulation object's plan go to the console in its Information dialog box.

If the notification level for a console message is lower than the notification level configured for the sim engine or the target simulation object, the message is not displayed. For example, if a simulation object's notification level is set to Info, it receives Error, Warn, and Info messages, but not messages set to Verbose or Debug.

➤ **To send a console message from a plan:**

1. In a plan window, choose **Commands > Console Message**. The Console Message dialog box opens.
2. Select the notify level from the list.
3. Type the message you want to send.
4. Click OK.

43.7.4 Creating Objects from Within a Plan

You can create any of the objects on the Create Object Palette from within a plan.

➤ **To create an object from a plan:**

1. In a plan window, choose **Commands > Create >** *palette*, where *palette* is one of the object creation palettes. The selected palette opens.
2. Select the object you want to create.
3. Create the desired object as you would if you were creating it directly from one of the object creation palettes. When you are finished creating the object, a Create command is added to the plan and the object itself disappears. (The object disappears because you are not actually creating the object right now, you are just creating the command that will create it when the plan executes.)

43.7.5 Deleting Objects from Within a Plan

You can delete any object in a scenario from a plan.

➤ **To delete an object from a plan:**

1. In a plan window, choose **Commands > Delete Object**. The Delete Object dialog box opens.
2. Type the name of the object you want to delete or select the object in the list or on the terrain.
3. Click OK.

Deleting One's Self from the Scenario

The Delete Object global command lets you delete any specified object. If you just want to delete the simulation object whose plan you are editing, you can use the Delete Self command. Like the Use Self option in some conditions, Delete Self provides a non-entity-specific command that is useful for plans that you want to assign to multiple simulation objects.

➤ To assign the Delete Self command, choose **Commands > Delete Self**.

43.7.6 Issuing a Plan

The Issue Plan global command lets you assign a plan to a simulation object. When you issue a plan, the new plan replaces whatever plan or task the simulation object is executing. Although you can issue a plan to any simulation object, this command is particularly useful when you create a simulation object after a scenario has started.

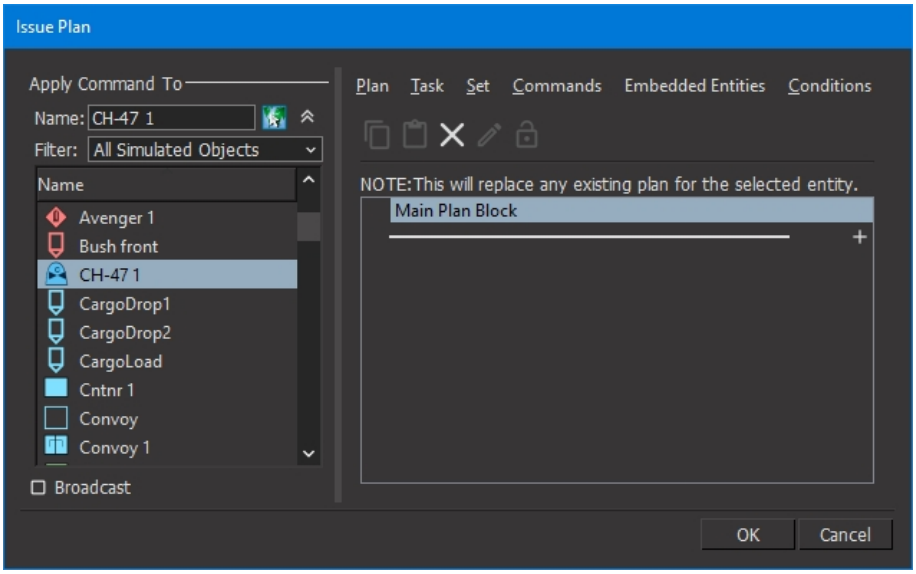
When you create a scenario, you cannot create a plan for a simulation object that does not exist. Therefore, if you create a simulation object after the scenario starts (using the Create global command), it is difficult to send it a sequence of tasks because global commands are not executed in sequence (as discussed in "42.1. Introduction to Individual Plans and Global Plans" on page 932). The Issue Plan command solves this problem.

Issue Plan is a concurrent task. Including it in a plan does not affect the current task that the simulation object is executing or the sequencing of tasks in its plan.

➤ **To issue a plan to a simulation object:**

1. In a plan window, choose **Commands > Issue Plan**. The Issue Plan dialog box opens (Figure 332).

Figure 332. Issue Plan dialog box



2. In the Apply command To: group box, select the simulation object to which you want to assign this plan or type the name of the simulation object in the Name text box.

   If this is a simulation object that does not yet exist (because you plan to create it after the scenario starts), you must type the name and you are responsible for ensuring that the name you type matches the name of the simulation object that will be created. You must also ensure that the scenario does not have more than one entity with the same name.

   If you want to issue a plan to an entity that has just been created, you can use the Created Object plan variable to identify the object. For more information about the Created Object variable, see "42.3. Plan Variables" on page 943.

3. Use the right side of the dialog box to write a plan. It has the exact same set of menus and commands as the Plan dialog box. Follow the procedures described elsewhere in this chapter to write the plan.

4. Click OK.

43.7.7 Sending View Control Messages

You can send view control messages to set the view mode for observers in the simulation. This includes the observers in VR-Forces and in any other VR-Vantage applications in the exercise. (The VR-Forces GUI is built using the VR-Vantage toolkit and is, therefore, considered a VR-Vantage application.) When you send a command you can direct it to a specific observer or to all similarly named observers, for example, all Observer 1s.

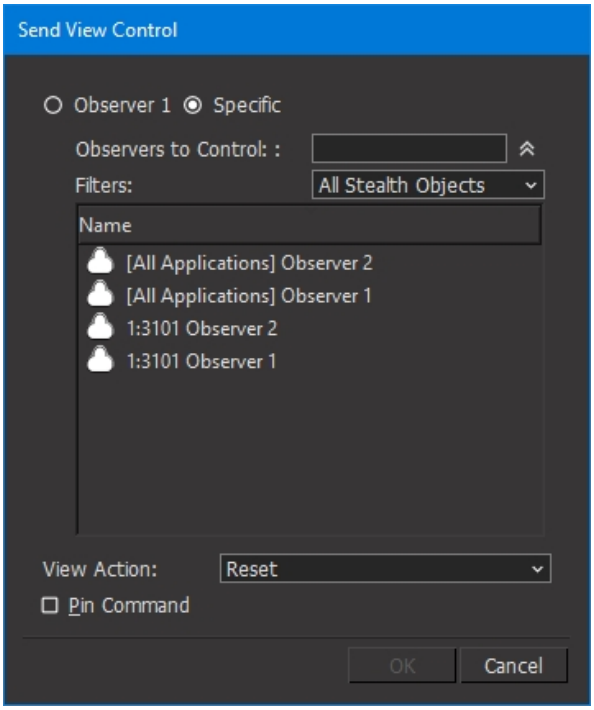
You can also send saved observer views. Observer views work only with VR-Forces. They do not affect other VR-Vantage applications. If you send an observer view, you can specify a transition time to the new view, similar to the automated transition described in "Animating Transitions Between Observer Views" on page 1204.

- The View Control command does not support all of the view modes. However, you can create a saved view using any view mode and send the saved view (to VR-Forces only).
- If you send an observer view, the current values for the observer view are put into the plan. If you subsequently change the observer view, the changed values are not picked up.
- If you record a scenario with the MAK Data Logger, it records view control commands. You can use them to create demonstration recordings that automatically change to different views as the recording progresses.

➤ **To send a view control message:**

1. In a plan window, choose **Commands > Send View Control**. The Send View Control window opens (Figure 333).
2. Select an Observers to Control option. Observer 1 sends the view control command to Observer 1 in all VR-Vantage applications. If you want to direct the view control message to a particular observer:
    a. Select Specific. The dialog box expands to show a list of observers.
    b. Select the observers you want to control by object name (such as 1:3101 Observer 1) or by observer name (such as Observer 2).
    c. Click OK.

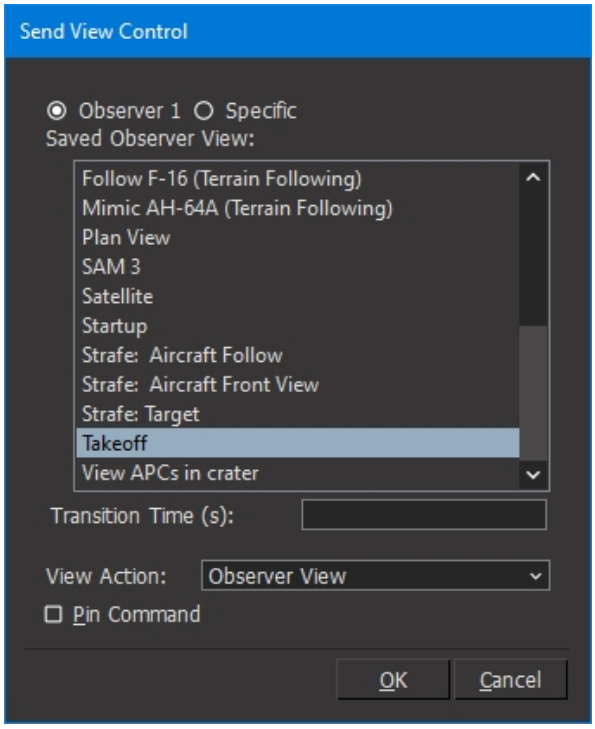
Figure 333. Observers to control



3. In the View Action list, select the view control you want to send:
   - Reset. Reset the observer view to the default view.
   - Detach. Detach the observer from an attached simulation object.
   - Observer View. Sends the selected observer view (from the Observer Views Panel).
   - *Attach mode*, where *Attach mode* is Absolute or one of the attach modes. Sends the selected attach mode.

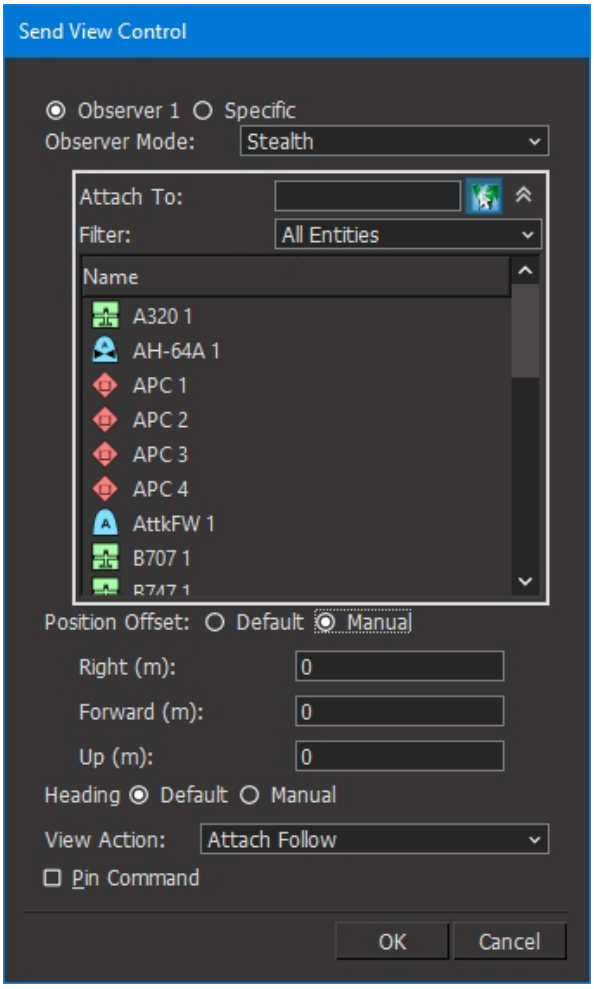
   For every command except Reset and Detach, the dialog box redisplays to add a selection window appropriate to the view mode.
4. If you selected Observer View as the View Action, select the observer view that you want to send (Figure 334). Optionally, specify a transition time, in seconds.

Figure 334. Observer view



5. If you selected Absolute mode or an attach mode as the View Action, complete the dialog box (Figure 335):
    a. In the Observer Mode list, select the observer mode for the view control message.
    b. If you selected Absolute mode, specify the location of the observer and, optionally, an offset orientation.
    c. If you selected an attach mode, select the simulation object to attach to.
    d. If you selected an attach mode that supports an offset vector or offset orientation, optionally select Manual and specify an offset. The default offset is 0,0,0.

Figure 335. Attach mode



e. If you select Track, specify the location for the observer to track from.

f. If you select Tether Track, select a secondary simulation object and, optionally, a position offset.

6. Click OK.

43.7.8 Creating a Sensor View from a Plan

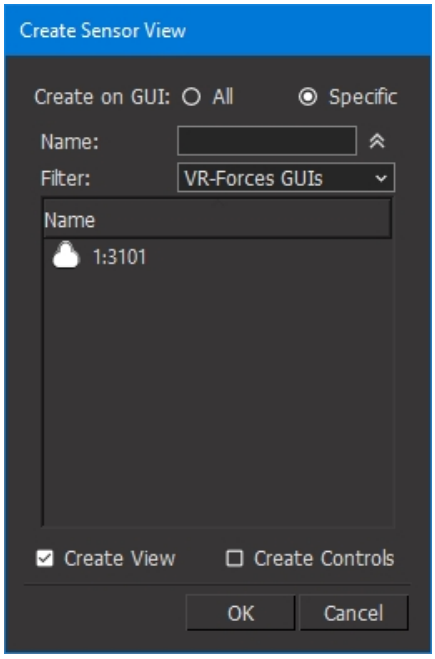
If a simulation object has a gimbaled sensor, it can create a view as part of its plan. You can specify whether the view displays in all GUIs or only a specific one. You can also indicate whether the view should include the sensor controls.

➤ **To create a sensor view from a plan:**

1. In a plan window, choose **Commands > Create Sensor View**. The Create Sensor View dialog box opens.
2. You can choose to create the sensor view in all GUIs or for a specific one. If you choose to create the view for a specific GUI, the dialog box allows you to specify which one.

3. Select the Create View check box to control whether the sensor view displays.
4. Optionally, select the Create Controls check box if you want to include view controls for the sensor view.
5. Click OK.

Figure 336. Create Sensor View



43.7.9 Sending Test Results

You can include test results in a plan for testing purposes. Test results print to the console, similar to console messages (as described in "43.7.3 Sending Console Messages" on page 972).

> You can set the sim to exit immediately after the test result is issued. For details, see the `--exitOnTestResultCommand` command-line option in Table 11 on page 178.

➤ **To send a test result from a plan:**

1. In a plan window, choose **Choose > Test Result**. The Test Result dialog box opens.
2. Select the result from the list.
3. Optionally, enter a message for the test result.
4. Click OK.

43.8. Writing Plans for Units

➤ **To write a plan for a unit:**

1. Select the unit.
2. Follow the procedures for writing a plan for an individual simulation object.

See "29.4.2 Independently Tasking Unit Members" on page 572 for notes about how simulation objects that are members of units respond to individual plans and independent tasks.

43.9. Writing a Plan for Multiple Simulation Objects

You can write a plan and assign it to multiple simulation objects at the same time. If any of the simulation objects already has a plan, it is overwritten by the new plan. After you assign a plan using this procedure, you can subsequently edit the plans for particular simulation objects and the changes do not affect the other simulation objects.

If you select several simulation objects and they all have the exact same plan, the plan window displays that plan. If any of the simulation objects have different plans, you are given the option to continue or cancel. If you continue, VR-Forces displays an empty Plan window and the new plan replaces whatever plans the simulation objects previously had.

➤ **To write a plan for multiple simulation objects:**

1. Select the simulation objects to which you want to assign the same plan.
2. Choose **Objects > Plan**. The Plan window opens.
3. Add statements to the plan as you would if you were writing a plan for an individual simulation object, as described elsewhere in this chapter.
4. Click OK. The plan is assigned to the selected simulation objects.

43.10. Writing Plans for Remote Entities

If you attach components to remote entities, you can write plans for those entities. To assign plans, the entities must exist in the exercise, must have attached components, and must be present in the exercise when VR-Forces saves the scenario. The plans must be appropriate for the attached components. Movement tasks are not appropriate, because VR-Forces cannot control the location of a remote simulation object. Appropriate plan statements might be to enable spot reporting, or to send a radio message. For details about how to attach components to remote entities, see "82.1. Attaching Components to Remote Entities" on page 1592.

43.11. Copying Plans and Plan Statements

You can copy plans and individual plan statements and paste them elsewhere in a plan or into another simulation object's plan. You can even paste them into another scenario if you do not close VR-Forces between the copy and paste operations.

> In addition to copy and paste, you can drag statements and condition blocks from one location to another in a plan and to other plans.

➤ **To copy plans and plan statements:**

1. Open the Plan window for the plan that you want to copy.
2. To copy the entire plan, select the top line in the plan (**Main Plan Block**). To copy a statement, select the statement. If you select a conditional statement, the entire statement block is selected.
3. Choose **Plan > Copy Plan** or click the Copy button ( ).
4. Open the Plan window for the simulation object you want to copy the plan to.
5. Select the line just above where you want to insert the copied plan.
6. Click the Paste button ( ).
7. Continue editing the plan.
8. Save the plan.

43.12. Restarting a Plan

There may be occasions when you want to restart a simulation object's plan without restarting a scenario. You can restart a plan interactively or by placing a Restart Plan command in a plan.

When you restart a plan, VR-Forces processes the statements in the plan, starting from the first statement. However, the simulation object executes tasks starting from its current location, not the location it had at the start of the scenario (unless it has not moved since the start of the scenario).

➤ **To restart a plan interactively:**

1. Select the simulation object whose plan you want to restart.
2. Choose **Objects > Manage Plan > Restart Plan**.

➤ **To restart a plan from within a plan:**

1. Open a simulation object's plan or a global plan.
2. Choose **Commands > Restart Plan**. The Restart Plan dialog box opens.
3. Select the simulation object whose plan you want to restart.
4. Click OK.
5. Save the plan.

43.13. Abandoning a Plan

If you give a simulation object that is executing a plan an independent task, it abandons the plan. You can also order a simulation object to stop executing its plan. When you abandon a plan:

- The simulation object stops whatever task it is working on, even if it is an independent task.
- The plan is marked as complete.
- All registered triggers are deleted.
- The statement indicator skips to the end of the plan.
- The simulation object enters a wait state.

> If you give a simulation object that is executing a plan an independent task, VR-Forces prompts you before it abandons the plan. (There is no prompt for the Abandon Plan command.) You can disable the prompt. For details, see "Enabling and Disabling the Task Confirmation Prompt" on page 561.

You can abandon a plan interactively or by sending an Abandon Plan command in an individual or global plan.

➤ **To abandon a plan interactively:**

1. Select the simulation object whose plan you want to abandon.
2. Choose **Objects > Manage Plan > Abandon Plan**.

➤ **To abandon a plan from within a plan:**

1. Open a simulation object's plan or a global plan.
2. Choose **Commands > Abandon Plan**. The Abandon Plan dialog box opens.
3. Select the simulation object whose plan you want to abandon.
4. Click OK.
5. Save the plan.

43.14. Editing a Plan During a Simulation

When the simulation is not running, you can make changes to the plan. Once you save the changes to the plan, the entity returns to the first line of the plan and starts it from the beginning. This can cause its activities to be out of synchronization with the other simulation objects in the simulation.

By default, plans are locked when you run a scenario. If you want to edit a plan while the simulation is running, you must click the Unlock button (). VR-Forces displays a warning message that changing the plan while the scenario is running can have unintended consequences.



44. Synchronization Matrix

This chapter describes the use of the Synchronization Matrix.

44.1. Overview of the Synchronization Matrix 984
44.1.1 How the Synchronization Matrix Works 984
44.1.2 The Synchronization Matrix Is an Object 985
44.1.3 Units and the Synchronization Matrix 985
44.1.4 Spot Report Viewpoint in the Synchronization Matrix 985
44.2. Building a Synchronization Matrix 986
44.2.1 Opening and Creating a Sync Matrix 986
44.2.2 Adding Entities and Units 987
44.2.3 Adding and Editing Phases 987
44.2.4 Adding and Editing Plans 988
44.2.5 Copying and Pasting Plans 989
44.2.6 Continuing a Plan 989
44.2.7 Deactivating and Activating Plans 990
44.2.8 Removing Plans 990

44.1. Overview of the Synchronization Matrix

The VR-Forces Synchronization Matrix (also called a sync matrix) allows you to work with multiple entities or units and add coordinating plans that are laid out in a series of phases. It is easier for you to create and manage scenarios where there is synchronization between entities or units. The intention is that you use one matrix for entities or units that are working together, rather than for all simulation objects or forces in the scenario. For example, the friendly and opposing forces would each have a separate matrix. If two groups in the same force are not coordinating their actions, they could also have separate matrices. An entity or unit can only be in one matrix in the scenario.

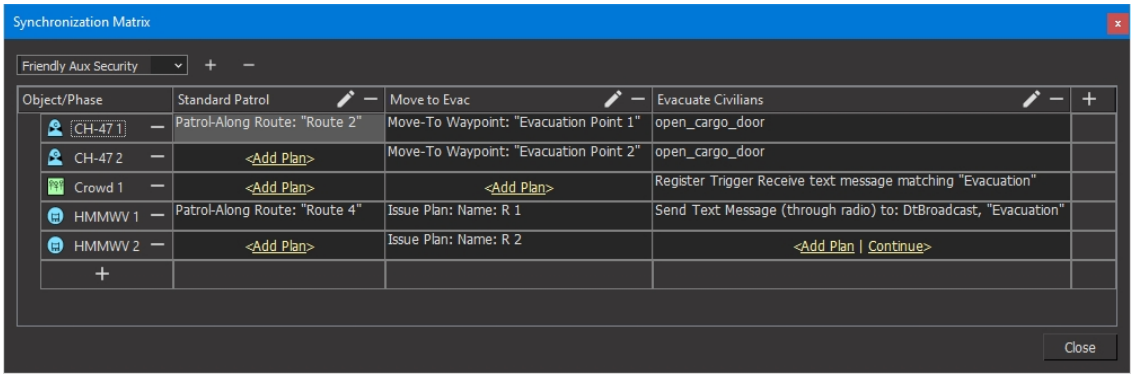
In scenarios where there is a high level of complexity, it can be hard to track down a problem where things are not happening as expected. By putting the plans for cooperating entities or units in a matrix, it is easier to see the relationships between their actions.

> ⚠ The Synchronization Matrix is another place for plans to exist. This can make it difficult to debug a scenario that includes individual plans on entities or units, a global plan, and also a collection of simulation objects with a Synchronization Matrix.

With individual plans, you can have a plan that has multiple branches based on the situation. The Synchronization Matrix, however, is intended for a linear series of actions.

You build a matrix for a particular scenario or battle plan. It is not a generic behavior that you can duplicate to another scenario or to another group of entities in the same scenario.

Figure 337. Synchronization Matrix Example



44.1.1 How the Synchronization Matrix Works

A VR-Forces Synchronization Matrix is a simulation object that uses the Issue Plan command to send the plans in the matrix to the entities or units. As a phase progresses, each entity and unit in the matrix receives its plan. The plan can be a simple task or it can be more complex. For example, you can set ordered speed and start an entity moving in the same phase. You could also have a reaction where an entity or unit detects an entity, moves to attack, and returns.

Each phase has a Phase End Condition, and when that condition is reached then the phases ends and the next phase begins. When a phase reaches its completion, all plan information from that phase is discarded and the plans in the next phase are used. See "44.2.3 Adding and Editing Phases" on page 987 for instructions.

44.1.2 The Synchronization Matrix Is an Object

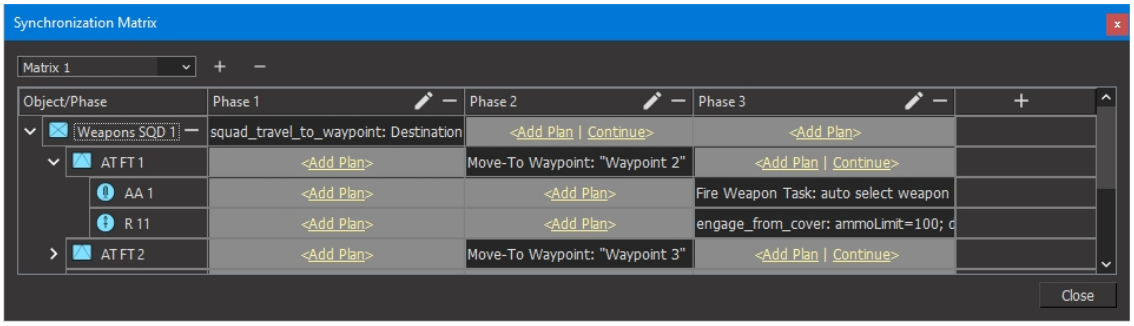
The Synchronization Matrix is a VR-Forces object and it exists on a single sim engine. A synchronization matrix can be on a different sim engine than other simulation objects. If the scenario has multiple matrices, you could put each matrix on a separate sim engine.

The Synchronization Matrix sends the issue plan commands, runs conditional tests, and monitors for phase completion.

44.1.3 Units and the Synchronization Matrix

A synchronization matrix can contain both entities and units. (It can also include part of a unit if that is appropriate for your scenario.) You can include a multi-level unit in a matrix. In the matrix, you can expand a unit to see the sub-units or entities. In each phase, you can make a plan for a unit or the entities that comprise the unit. You cannot do both in the same phase.

Figure 338. Synchronization Matrix for Multi-Level Unit



44.1.4 Spot Report Viewpoint in the Synchronization Matrix

When you configure VR-Forces to show the ground truth for only some of the forces, objects in the other forces are hidden from the Synchronization Matrix unless they are spotted.

These figures show the Synchronization Matrix when Ground Truth for Opposing force is disabled. In Figure 339, the Friendly force has not detected any members of the Opposing force. In Figure 340, the Friendly force has spotted (and destroyed) some members of the Opposing force.

Figure 339. No contacts

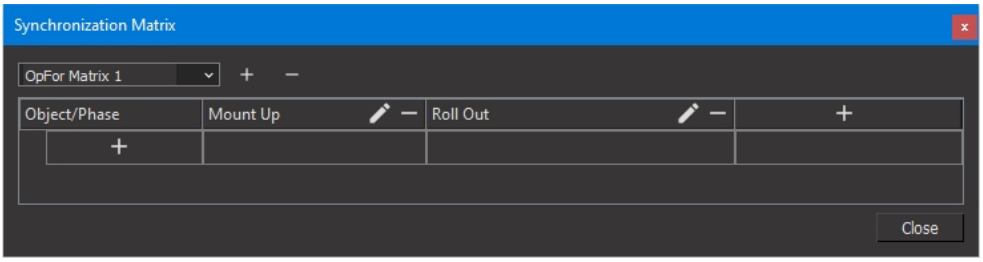
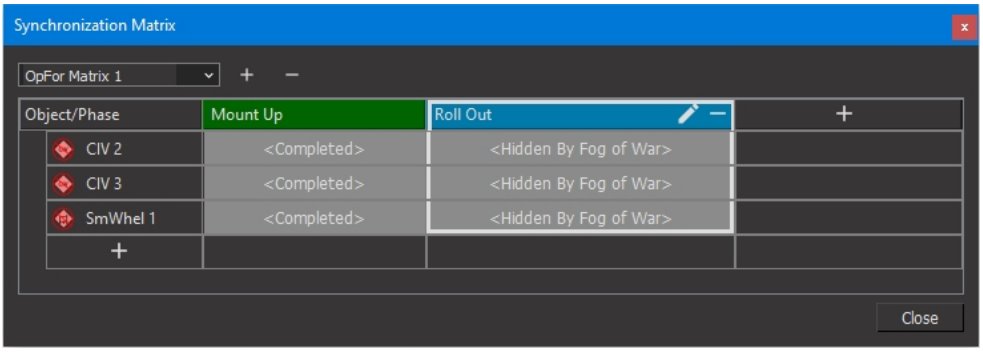


Figure 340. Some contacts



See "9.1.2 Configuring the Spot Reports Viewpoint" on page 291 for details on spot reports and fog of war.

44.2. Building a Synchronization Matrix

Building a Synchronization Matrix involves:

1. Creating a matrix.
2. Adding entities and/or units to the matrix.
3. Adding phases to the matrix.
4. Adding plans for the entities and units.

44.2.1 Opening and Creating a Sync Matrix

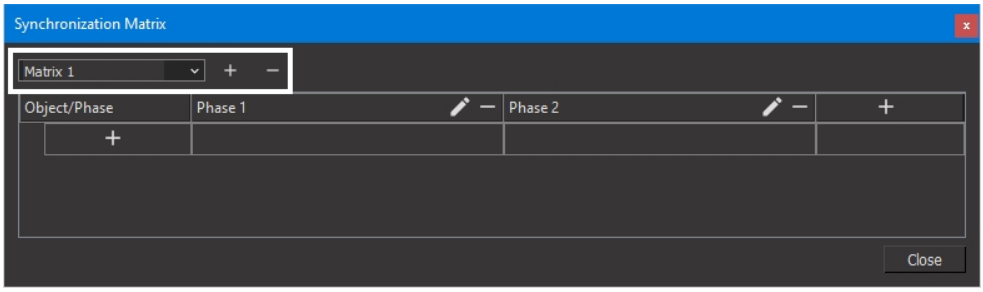
The Synchronization Matrix dialog box (Figure 341) allows you to manage each sync matrix in your scenario. You use it to add entities or units to the matrix, manage their plans, and add and edit the phases in the matrix.

When you open the Synchronization Matrix for the first time in a scenario, it is blank. If you have one or more matrices in the scenario, the first matrix displays.

➤ **To open the matrix, do one of the following:**

- Choose **Simulation > Synchronization Matrix**.
- Click the Open Synchronization Matrix button (#) in the Scenario Components toolbar or the Last Selected Object Panel.

Figure 341. New Synchronization Matrix



➤ To create a new matrix, click the Add Sync Matrix button (+).

➤ To view or edit a matrix, select it from the list.

➤ To change the name of the matrix, simply type in the text box.

➤ To delete the currently selected matrix, click the Remove Sync Matrix button (−).

44.2.2 Adding Entities and Units

To create the Synchronization Matrix, you need to add the entities or units that will be working together. The matrix can include entities and units.

➤ **To add an entity or unit to the matrix:**

1. In the Object column, click the Add Object button (+). The cursor turns to a hand.
2. Select the entity or unit in the Objects List or on the terrain. It is added to the matrix.

You can also remove an entity or unit from the matrix.

➤ To remove the object, click the Remove Object button (−) next to the entity or unit's name.

44.2.3 Adding and Editing Phases

A new Synchronization Matrix starts with two phases. You can add more phases as needed.

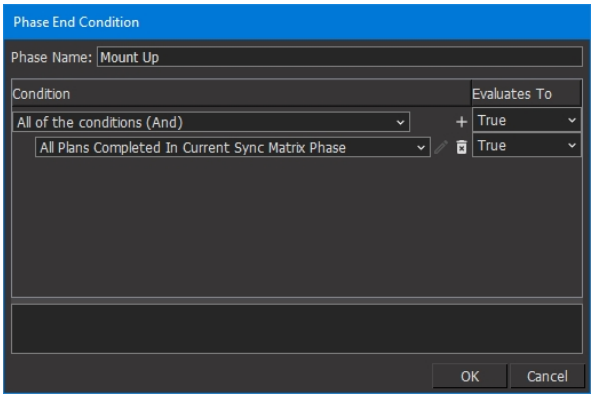
➤ To add a phase, click the Add Phase button (+) at the top of the right-most column of the matrix.

You can edit a phase to specify the phase end condition. The default end condition is All Plans Completed in Current Sync Matrix Phase. In that case, the next phase begins only when all entities and units in the matrix have completed their plans in that phase. You can also specify another condition. See "42.2. Conditional Statements" on page 933 to understand more about conditions. When the phase end condition is met, the Synchronization Matrix moves to the next phase.

➤ **To edit a phase:**

1. In the header for the phase, click the Edit button ( ). The Phase End Condition dialog box opens (Figure 342).

Figure 342. Phase End Condition



2. Change the condition to meet your needs. See "43.4. Building Condition Statements" on page 956 for more information.
3. If desired, change the Phase Name.
4. Click OK. You return to the Synchronization Matrix.

You can also remove a phase from the matrix.

➤ To remove the object, click the Remove Phase button ( ) next to the phase name and then click Yes in the confirmation message.

44.2.4 Adding and Editing Plans

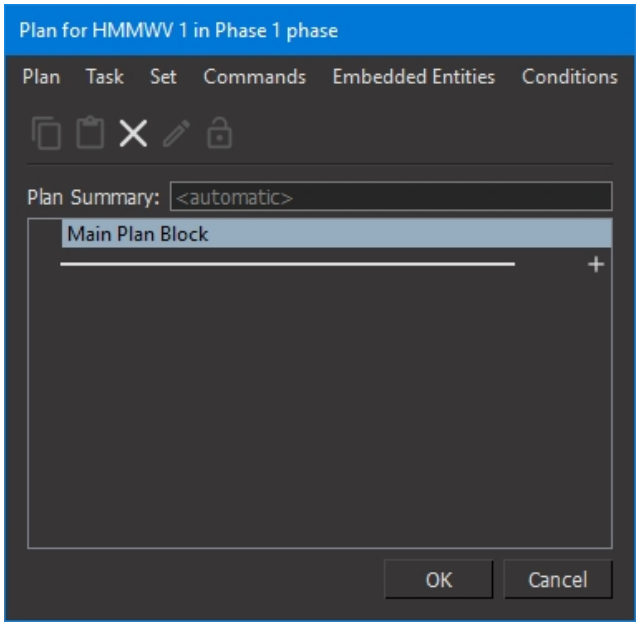
Each cell in the Synchronization Matrix can hold a plan for the associated entity or unit.

➤ **To add or edit a plan, do one of the following:**

- Single-click in a cell that reads Add Plan.
- Right-click in a cell and choose **Set Plan** or **Edit Plan**.

The Plan for Phase dialog box opens (Figure 343).

Figure 343. Plan for Phase dialog box



Use the Plan for Phase dialog box to create the entity or unit's plan. See "Chapter 43. Writing Plans" on page 947 for more details.

The Plan for Phases includes a Plan Summary box where you can enter a description of the plan to display in the Sync Matrix.

44.2.5 Copying and Pasting Plans

You can copy a plan from one cell in the Synchronization Matrix and paste it in another.

➤ **To copy and paste a plan:**

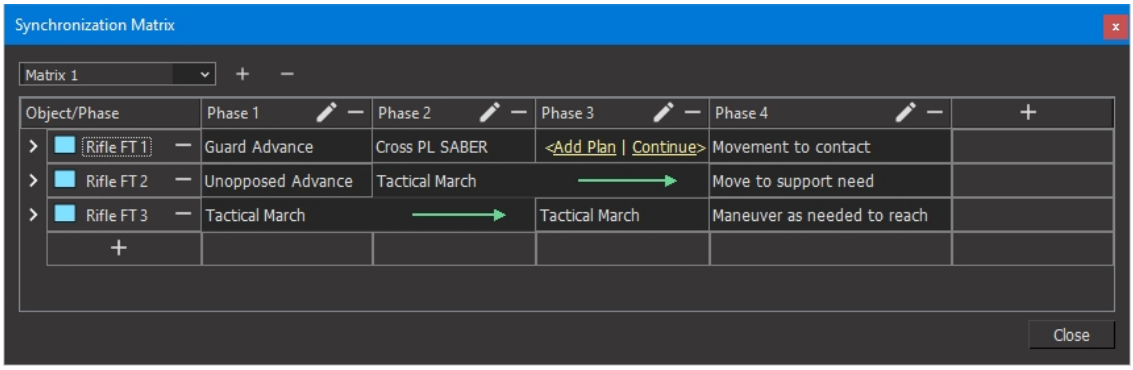
1. Right-click on a cell with a plan and then choose **Copy Plan**.
2. Right-click on another cell and the choose **Paste Plan**.

44.2.6 Continuing a Plan

Sometimes, you might want an entity or unit's plan to continue across two or more phases. This allows one phase to end and the next phase to begin while the entity or unit continues to perform its plan, rather than abandoning if there is no plan set. A phase with a plan marked to continue does not prevent the sync matrix from continuing to the next phase. Once the sync matrix reaches the last phase where the plan is continued, it checks whether the plan is fully complete as part of the phase end condition test.

➤ To continue a plan through a phase, click Continue in the cell for that phase. An arrow displays, showing that the plan spans that phase as well as the phase where it begins.

Figure 344. Synchronization Matrix with continued plans



You can also discontinue a plan that was previously set to continue through a phase.

➤ To remove the continuation of a plan, right-click on the arrow and then choose **Discontinue Previous Plan**.

44.2.7 Deactivating and Activating Plans

You can deactivate plans for units. When a unit's plan is deactivated, the subordinates perform their own plans.

➤ To deactivate a plan, right-click on the plan for a unit and then choose **Deactivate Plan**.

You can also reactivate a plan that you had previously deactivated.

➤ To reactivate a plan, right-click on the unit's plan and then choose **Reactivate Plan**.

44.2.8 Removing Plans

➤ To remove a plan, right-click on a plan and then choose **Remove Plan**.



IX. Tactical Graphics

Tactical graphics are objects that represent points, lines, areas, and information overlaid on the terrain. This section describes overlays and particular details about tactical graphics.

- For generic information about creating tactical graphics, see "Chapter 14. Creating and Placing Objects" on page 371.
- For information about creating and using spawn points, sink points, and air traffic areas (all of which are tactical graphics), see "Chapter 16. Generating Traffic (Pattern of Life)" on page 415.
- For information about combat engineering objects, which are displayed similarly to tactical graphics, see "27.2. Combat Engineering Objects" on page 533.



45. Introduction to Tactical Graphics

This chapter is a brief introduction to tactical graphics.

45.1. Overview 994
45.2. Points 996
45.3. Phase Lines 996
45.4. Routes 997
45.5. Areas 997
45.6. Obstacles 997
45.7. Displaying Tactical Graphics 998
45.7.1 Showing and Hiding Tactical Graphics 998
45.7.2 Showing and Hiding Tactical Graphics Labels 1000
45.7.3 Displaying HAT Lines for Vertices 1000
45.7.4 Displaying HAT Lines for Routes (3D Only) 1001
45.7.5 Displaying Tactical Graphics as Black or White 1001

45.1. Overview

You can add tactical graphics to scenarios. Tactical graphics are objects that represent points, lines, areas, and information overlaid on the terrain. They are automatically assigned to overlays. (You can change the overlay at or after creation.) Simulation objects have knowledge of some types of tactical graphics, which allows tasks and plans to take them into account. VR-Forces supports these types of tactical graphics:

Table 43. Types of tactical graphics

Tactical Graphic	Description
Control objects	**Simulation objects are aware of control objects and can respond to them. They avoid obstacles, can move to waypoints, follow routes, and so on.**
Waypoint	A specific X, Y, Z coordinate.
Area	A polygonal area. Areas are considered to have volume, so that air and subsurface entities can interact with them.
Route	A sequence of points in three-dimensional space.
Phase Line	Infinitely long line defined by two or more points.
Obstacle	A polygonal area on the terrain that simulation objects try to avoid if obstacle avoidance is enabled.
Linear Obstacle	A line on the terrain that simulation objects try to avoid if obstacle avoidance is enabled. Linear obstacles are supported only for entity-level scenarios. They work only in navigation areas.
Pedestrian Path	When humans plan paths in navigation areas that have navigation data, if a pedestrian path is present that can take them to their destination, they will use it. You can think of a pedestrian path as a route that a human will automatically choose to follow instead of being told to follow it. Pedestrian paths are supported only for entity-level scenarios.
Terrain Page-in Area	An area in which streaming or paged terrain is given priority for paging in.
Other tactical graphics	
Terrain Profile Line	A line that lets you examine the profile of the terrain along the line.
Intervisibility Line	A line that shows line-of-sight between two points.
Lines	Generic lines and lines with particular graphics, such as main axis and fortified line.
Bullseye	A graphic for measuring distances between a simulation object or the mouse cursor and the bullseye.
Points	Various military symbology, such as point of contact and decision point. Simulation objects treat most points as waypoints.

Table 43. Types of tactical graphics (continued)

Tactical Graphic	Description
Areas	Minefields and other areal objects. These are not treated like area control objects.
Weather	A variety of weather conditions that display in the 3D view. For information about weather tactical graphics, see "11.6. Using Weather Simulation Objects" on page 339.
Text	Any arbitrary informational text on the screen.
Symbols	Symbols that do not represent simulated objects. They are just graphics on the screen and do not affect the simulation.
Circles, ellipses, boxes, and spider webs	These are only graphics and do not affect the simulation.
Volumetric objects	Volumetric areas, circles, boxes, ellipses, and corridors allow you to test for a simulation object's presence in a 3D volume rather than just a 2D area. They are useful for defining air corridors and airspace.
Combat engineering objects	Combat engineering objects are lines and areas that can impede the progress of simulation objects and cause damage to them. Combat engineering objects are supported only in aggregate-level scenarios. For information about how they work, see "27.2. Combat Engineering Objects" on page 533.

In aggregate-level scenarios, linear obstacles and pedestrian paths are not supported.

Tactical graphics have names. New objects are assigned default names in the format **Point 1**, **Point 2**, **Route 1**, **Route 2**, and so on. You can specify an alternative name when you create the object. A graphical object's name can be up to 255 characters long.

Tactical graphics can have labels and extended labels. They do not have to be unique and do not have a length restriction.

Tactical graphics are assigned to a force. Waypoints and bullseyes are displayed using their force color. To determine the force of other tactical graphics, examine the Force attribute in the State Data page of the graphic's Information dialog box. For more information, see "21.1. Displaying Information About an Object" on page 462.

The following sections provide more information about the basic control objects. For information about the other tactical graphics types mentioned in this section, see:

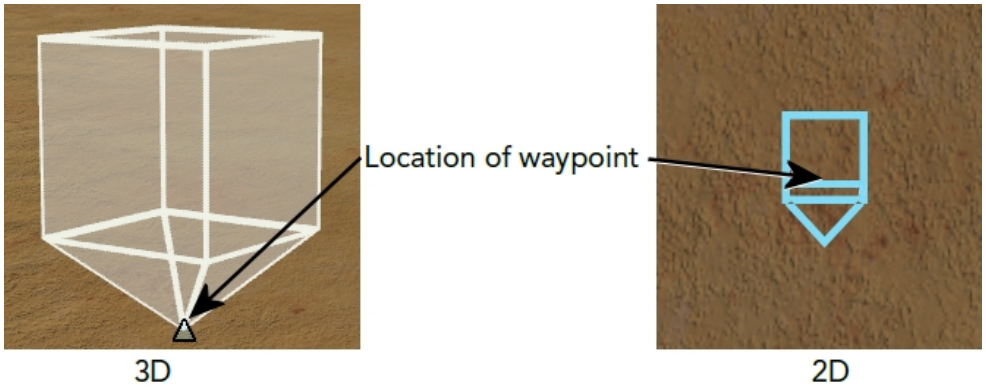
- "Chapter 55. Intervisibility and Range" on page 1121.
- "65.1. Terrain Profiles" on page 1266.
- "Manually Paging-In Terrain in VR-Forces" in the *Adding Content to MAK ONE Applications Reference Manual*.
- "Chapter 14. Creating and Placing Objects" on page 371.
- "Chapter 46. Overlays" on page 1003.

45.2. Points

A point identifies an X, Y, Z coordinate in the terrain. You can command a simulation object to travel to a point and to travel back and forth between points. Figure 345 illustrates waypoints in the 3D and 2D views. In 3D, the bottom of the symbol points to the location of the waypoint. In 2D, it is centered. You can place several different kinds of points on overlays.

A point can have an altitude value. This allows you to send aircraft to waypoints or have them patrol between points.

Figure 345. Waypoint symbol



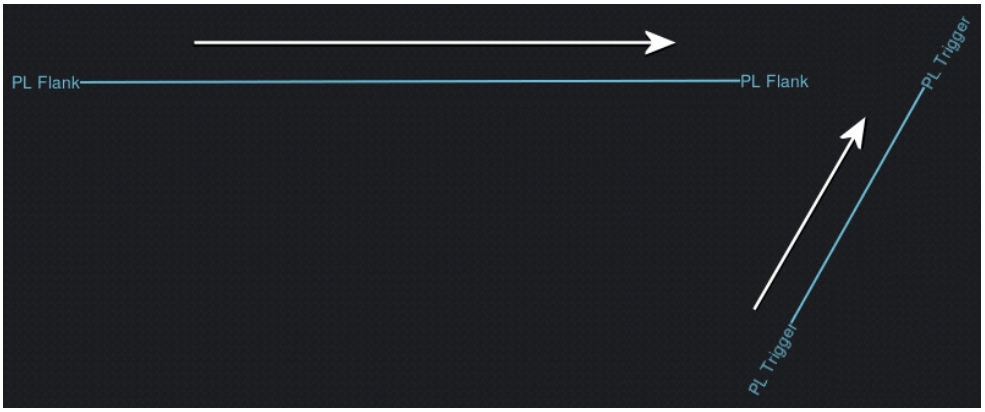
Waypoints clamp to the actual surface of the entity. To place a waypoint on a simulation object in the ocean, you may need to disable buoyancy before placing the waypoint (or other point tactical graphic); see "53.3. Enabling Buoyancy for Surface Entities" on page 1111.

45.3. Phase Lines

A phase line is defined by two or more points, but is considered to extend indefinitely in either direction and have infinite height. A phase line is used to test the location of a simulation object, for example: Is simulation object A to the left of phase line B? The left side of the line is determined relative to the direction in which the line is drawn. The phase line's labels are placed at the beginning and end of the line. Figure 346 illustrates two phase lines. The arrows are placed to the left of the line and indicate the direction of the line.

Simulation objects cannot move along phase lines. Do not use a phase line if you want a two-point route.

Figure 346. Phase line direction and left side



45.4. Routes

A route is a sequence of points in three-dimensional space. Like phase lines, routes have a direction. A line's name is displayed at its beginning.

Simulation objects can move along routes from one end to the other, and can patrol back and forth along them.

Do not create a two-point route if you want a phase line. VR-Forces cannot determine whether a simulation object is to the left or right of a route.

45.5. Areas

An area defines a polygonal area on the terrain. (This is not a terrain polygon; it is only a graphic.)

Areal tactical graphics represent a distinct place or an area with a specific purpose. Areas can also be used to test the location of a simulation object, for example: Is simulation object A in area B?

45.6. Obstacles

Obstacles represent lines and areas of the terrain that would cause a simulation object problems, such as a ditch or minefield. In entity-level scenarios, if obstacle avoidance is enabled, if a ground entity encounters an obstacle in its path, it tries to drive around the obstacle.

In addition to obstacles, aggregate-level scenarios support combat engineering objects and contamination areas. These objects can defend, damage, or impede simulation objects.

The many combat engineering objects available in aggregate-level scenarios make simple obstacles comparatively less useful. Linear obstacles are not supported in aggregate-level scenarios. They only work in navigation areas.

45.7. Displaying Tactical Graphics

You can configure the display of tactical graphics:

- Enable and disable the display of tactical graphics.
- Show or hide vertex indicators for point, linear, and areal objects. (By default, vertex indicators are displayed only when a tactical graphic is selected, except for intervisibility and range lines.)
- Show or hide height-above-terrain lines.
- Display height-above-terrain lines only for selected graphics.

Figure 347. Tactical graphics with height-above-terrain lines (3D)

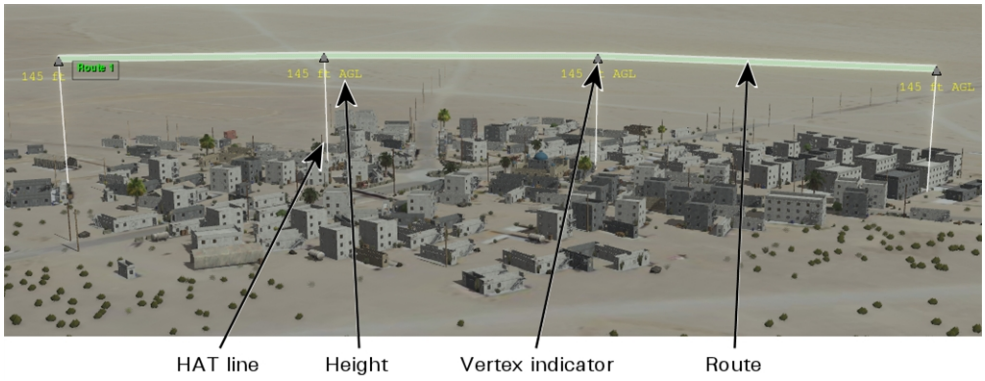


Figure 348. Tactical graphic (2D)



45.7.1 Showing and Hiding Tactical Graphics

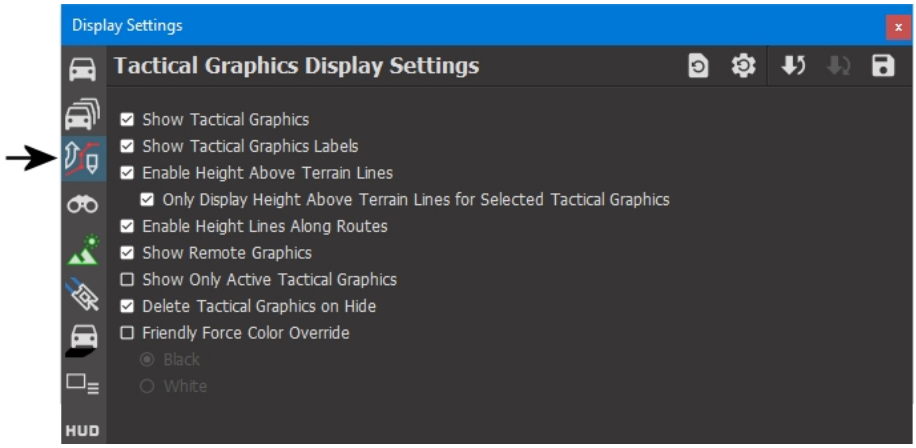
You can control the display of tactical graphics from the Display Settings dialog box, Tactical Graphics Display Settings page, from menu and toolbar options, and from the Overlays tab.

By default, tactical graphics are destroyed when you hide them and recreated when you show them again. This can cause performance issues when showing and hiding tactical graphics. See "Configuring How Tactical Graphics Are Hidden" on the next page.

➤ **To show or hide tactical graphics:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349).

Figure 349. Tactical Graphics Display Settings page



3. Select or clear the Show Tactical Graphics check box.
4. To show only active tactical graphics, select Show Only Active Tactical Graphics. For more information about active tactical graphics, see "47.11. Active and Inactive Tactical Graphics" on page 1035.

- You can also enable and disable tactical graphics by clicking the Show/Hide Tactical Graphics button ( ) on the Display Settings Toolbar.
- You can choose whether to display tactical graphics only when they are active by clicking the Show Only Active Tactical Graphics button ( ).

Showing and Hiding Tactical Graphics on the Overlays Tab

On the Overlays tab, you can show and hide tactical graphics individually and by overlay.

➤ **To control the display of tactical graphics from the Overlays tab:**

1. On the Objects List Panel, select the Overlays tab (Figure 351 on page 1005).
2. To hide all graphics on a particular overlay, clear the check box for that overlay.
3. To hide a specific tactical graphic, clear its check box.
4. To redisplay a hidden tactical graphic or overlay, select its check box.

Configuring How Tactical Graphics Are Hidden

You can make tactical graphics invisible instead of destroying them when you choose to hide tactical graphics. In scenarios with a number of tactical graphics that you want to hide and display again, this can improve performance.

To make tactical graphics invisible when hidden:

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349 on the previous page).
3. Select or clear the Delete Tactical Graphics on Hide check box.

45.7.2 Showing and Hiding Tactical Graphics Labels

In addition to displaying the outlines of tactical graphics, you can display labels for these objects. For example, labels are enabled in Figure 347 on page 998 and Figure 348 on page 998. VR-Forces can display a tactical graphic's label and height above terrain (Figure 349 on the previous page).

Displaying vertex labels can affect performance.

➤ **To show or hide vertex labels:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349 on the previous page).
3. Select or clear the Show Tactical Graphics Labels check box.

- You can also enable and disable tactical graphics labels by choosing **Settings > Tactical Graphics Labels** or clicking the Tactical Graphics Labels button () on the Display Settings Toolbar.
- Text on tactical graphics is hidden when the observer is at an altitude such that the text is longer than the adjacent line segment or shape. Because of this, the text on larger tactical graphics is visible at higher altitudes than the text on smaller ones.

45.7.3 Displaying HAT Lines for Vertices

In 3D observer modes, you can display height-above-terrain (HAT) lines for the vertices of tactical graphics (Figure 347 on page 998). These vertical lines between an object's vertices and the ground help you understand the relationship of the object to the terrain. The height for each vertex displays (using whatever units are configured on the Display Units Settings page).

HAT lines at vertex points are shown only for tactical graphics that could be above or below the ground, such as routes and waypoints. You can show and hide these lines and you can specify that they be shown for all objects or only for the selected objects.

> ⚠ Use of height above terrain lines can reduce the frame rate.

➤ **To show or hide height-above-terrain lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349 on page 999).
3. To display HAT lines, select the Enable Height Above Terrain Lines check box.
4. To only display lines for selected objects, select the Only Display Height Above Terrain Lines For Selected Tactical Graphics check box.

45.7.4 Displaying HAT Lines for Routes (3D Only)

You can choose whether to display HAT lines for route tactical graphics in the scenario. These thin, light-colored lines display in regular intervals, pointing towards the terrain to help you visualize the height. There is no terrain intersection; if the route is above water, the HAT lines can go below sea level.

> ⓘ To configure the appearance of the HAT lines, edit the Environmental Line visual definition for the route tactical graphic's element definition and specify the values for **enableheightlines**, **heightlinescolor**, and **heightlinesspacing**. See the *Element Definitions* chapter in the *Adding Content to MAK ONE Applications Reference Manual* to learn more.

➤ **To enable or disable HAT lines for track histories:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349 on page 999).
3. Select or clear the Enable Height Lines Along Routes check box.

45.7.5 Displaying Tactical Graphics as Black or White

By default, most tactical graphics display as the color of the selected force. You can also change the color of tactical graphics using the create/edit dialog box. In the display settings, you can override these colors for friendly force tactical graphics to display them as either black or white.

➤ **To override the colors of friendly force tactical graphics:**

1. Choose **Display > Settings**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 349 on page 999).
3. Select the Friendly Force Color Override check box.
4. Select either the Black or White option.



46. Overlays

This chapter explains how to create and edit overlays.

46.1. Creating and Editing Overlays 1004
46.1.1 Creating an Overlay 1005
46.1.2 Selecting Overlays 1005
46.1.3 Locking an Overlay 1005
46.1.4 Hiding the Objects on an Overlay 1006
46.1.5 Hiding Newly Created Objects 1006
46.1.6 Changing an Overlay's Name 1007
46.1.7 Removing an Object from a User-Created Overlay 1007
46.1.8 Deleting an Overlay 1007
46.2. Saving and Loading Tactical Graphics and Overlays 1007
46.2.1 Loading Tactical Graphics and Overlays 1008
46.3. Exporting Tactical Graphics as Shapefiles 1008
46.3.1 Importing Shapefiles into Overlays 1009

46.1. Creating and Editing Overlays

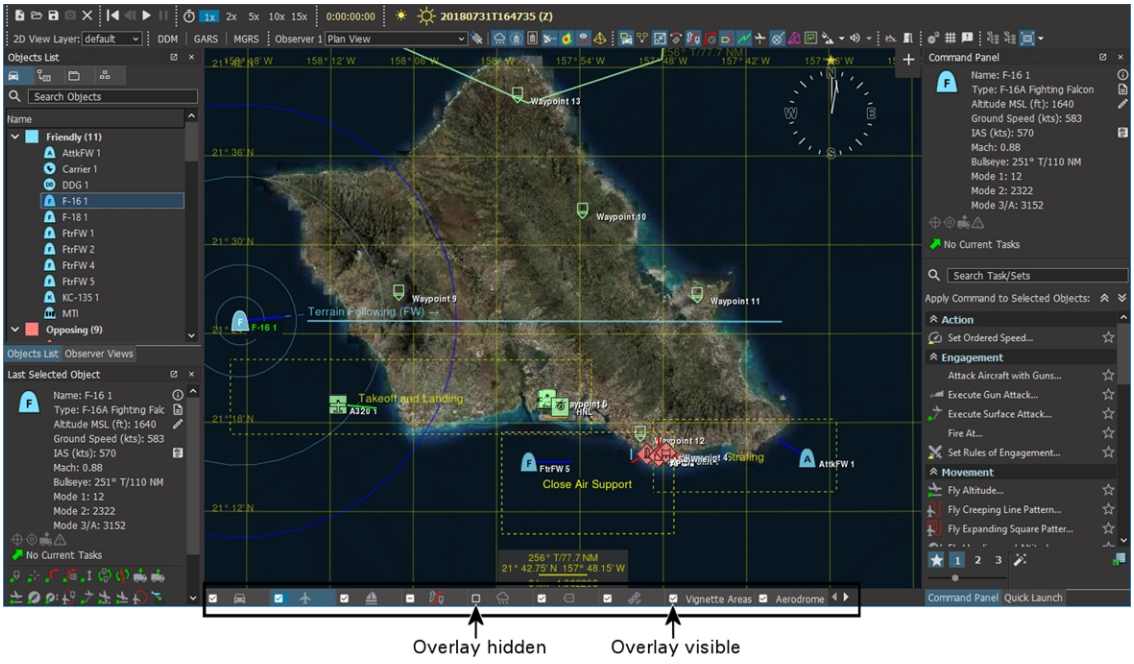
Overlays provide a convenient way to organize and manage the tactical graphics and simulation objects in a simulation. They are the equivalent of adding clear plastic overlays to a physical map and drawing on them. You can add any number of overlays. You can enable or disable editing of the objects they contain.

When you create an object, if you do not explicitly assign it to an overlay, it is placed on a default overlay. By default, simulation objects are placed on an overlay that corresponds to its high level filter – ground, air, human, and so on. All tactical graphics are placed on the tactical graphics overlay.

You can add overlays in the Simulation Object Editor and specify that they be used as the default overlay for specific object types. They are identified by text. You cannot specify an icon.

Overlays and their contents are displayed in the Overlays View of the Objects List Panel. The default overlays are identified by icons on the Overlay toolbar and by icons and text on the Overlays View of the Objects List Panel. You can show and hide objects by overlay and individually (Figure 350).

Figure 350. Overlays View and Overlay Bar

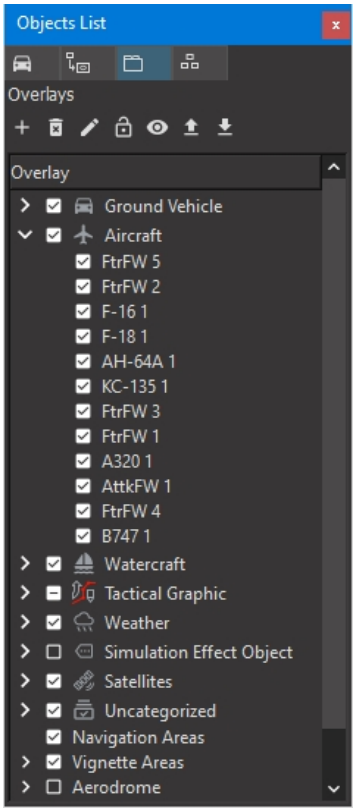


46.1.1 Creating an Overlay

➤ **To create an overlay:**

1. On the Objects List Panel, select the Overlays View (Figure 351).

Figure 351. Overlays View



2. Click the Add button (+). An overlay is added to the list with a default name.
3. Optionally, change the name as described in "46.1.6 Changing an Overlay's Name" on page 1007.

46.1.2 Selecting Overlays

➤ To select an overlay, click its name on the Overlays View.

46.1.3 Locking an Overlay

When an overlay is locked, you cannot edit the objects on it. You can still add and delete objects.

When you save a scenario, VR-Forces saves the locked/unlocked state of each overlay. (If you lock or unlock an overlay while you are running a scenario, but do not save the scenario, the state of the overlay is not saved.

➤ **To lock or unlock an overlay:**

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay you want to lock or unlock.
3. Click the Lock/Unlock Overlay button ( or ).

46.1.4 Hiding the Objects on an Overlay

You can hide the objects on an overlay. This may be convenient if you want to focus on a particular set of simulation objects or tactical graphics while you are editing or observing a scenario. You can hide all of the objects on an overlay or individual objects. You can also quickly:

- Hide all overlays except the selected one.
- Hide all overlays.
- Show all overlay.

When you save a scenario, the show/hide state of default overlays and objects on them is not saved. The state of user-created overlays is saved.

➤ To hide all of the objects on an overlay, on the Overlay Bar or on the Object Lists Panel, Overlays View, clear the check box for the overlay whose objects you want to hide.

➤ To hide individual objects on an overlay, on the Objects List Panel, Overlays View, clear the check box for the object.

➤ **To hide all overlays except the selected one:**

1. Right-click the overlay that you want to be visible. A menu opens.
2. Choose Hide All but This.

➤ **To show all overlays:**

1. Right-click any overlay. A menu opens.
2. Choose Show All Overlays.

➤ **To hide all overlays:**

1. Right-click any overlay. A menu opens.
2. Choose Hide All Overlays.

46.1.5 Hiding Newly Created Objects

You can specify that newly created objects be hidden. When you set this option on one of the default overlays, the setting is not saved. When you set this option on a user-created overlay, it saves with the scenario.

➤ **To specify that newly created objects be automatically hidden:**

1. On the Overlay Views panel, select the overlay whose objects you want hidden.
2. Click the Auto-hide button (). A crossed out eye () is added to the overlay name to show that objects created on this overlay are hidden.

46.1.6 Changing an Overlay's Name

You can customize the name of a default overlay, such as Ground Vehicle or Tactical Graphics. However, the next time you create an object that is assigned to a default overlay, unless you assign it to the renamed overlay, a new overlay with the default name is created.

➤ **To change an overlay's name:**

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay that you want to rename and click the Rename button, or slowly double-click the overlay whose name you want to change. The name becomes editable.
3. Type a new name.
4. Press **Enter**.

46.1.7 Removing an Object from a User-Created Overlay

You can remove objects from a user-created overlay. You might do this if you want to delete the overlay but do not want to remove the simulation object.

➤ To remove an object from a user-created overlay, in the Overlays View, right-click the object that you want to remove from the overlay and then choose **Remove From Overlay**.

46.1.8 Deleting an Overlay

➤ **To delete an overlay:**

1. On the Objects List Panel, select the Overlays View.
2. Select the overlay you want to delete.

   ⚠ When you delete an overlay, all of the simulation objects and tactical graphics on it are also deleted.

3. Click the Delete button ().

46.2. Saving and Loading Tactical Graphics and Overlays

If you publish a tactical graphic, it is automatically saved to the scenario objects file (*.oob*). If you do not publish a tactical graphic, it is automatically saved to the overlay file (*.ovl*). There may be cases where you want to create a set of overlays for use in multiple scenarios. You can save them to an overlay file independently of the scenario, or even save the overlays and not save the scenario at all.

> When you save overlay objects, only unpublished tactical graphics are saved.

To save overlays and tactical graphics:

1. Choose **File > Overlay > Save Overlay**. The Save Overlay dialog box opens.
2. Type a name for the overlay file.
3. Click Save.

46.2.1 Loading Tactical Graphics and Overlays

You can load overlays that you have saved from a previous session.

➤ **To load overlays:**

1. Choose **File > Overlay > Load Overlay**. The Load Overlay dialog box opens.
2. Select the overlay file you want to load.
3. Click Open.

> When you load overlay objects, you must use the same terrain database you created them on, or a terrain database that includes the area they were created in.

46.3. Exporting Tactical Graphics as Shapefiles

You can export point, line, and area tactical graphics to shapefiles. VR-Forces exports all of the graphics on a selected overlay. This may be useful if you need feature data for a terrain, such as a road network, but there is no feature data available from external sources. You can edit the shapefile created by VR-Forces in an appropriate third-party application to add the attributes needed to fully specify the features. For example, suppose you need a road network. You would create routes in the proper locations in VR-Forces and export the overlay to a shapefile. Then you would edit the shapefile to connect the road segments, specify road width, number of lanes, and so on. You could then load the shapefile in VR-Forces or any other application that supports shapefiles.

When VR-Forces creates shapefiles from tactical graphics, it creates separate files for points, lines, and areas. It appends *P.shp*, *L.shp*, or *A.shp* to the filename you provide based on the tactical graphics types being exported. For example, if an overlay has points and lines and you export them to myFeatures, VR-Forces creates files called *myFeaturesP.shp* and *myFeaturesL.shp*.

➤ **To export tactical graphics as shapefiles:**

1. On the Objects List Panel select the Overlays View.
2. Select the Overlay whose tactical graphics you want to export.
3. Choose **File > Overlay > Save Overlay as Shape Files** or click the Save Overlay as Shapefile button () on the Overlays View. The Save Overlay as Shape Files dialog box opens. (You can also right-click an Overlay on the Overlay Bar and open the context-sensitive menu.)

4. Type a name for the shapefile.
5. Click Save.

46.3.1 Importing Shapefiles into Overlays

You can import feature data into overlays. Features are placed as points, lines, or areas. This is a quick way to create tactical graphics for features such as roads, buildings, and so on. For example, if you were to load the *MAK Earth (online).mtf* terrain and import *BuildingsP.shp*, which is used in the Little Pond terrain, it places a group of waypoints at the locations of the buildings.

> Importing shapefiles into an overlay is not the same thing as adding a feature layer to the terrain. For example, importing a file such as *RoadsL.shp* to an overlay creates routes that can be referenced in movement tasks. Adding it as a feature layer creates a road network that could be used for path planning.

➤ **To import feature data into an overlay:**

1. Select an overlay.
2. Choose **File > Overlay > Import Shape File Onto Overlay** or click the Import Shapefile Onto Overlay button () on the Overlays View. The Import Shapefile Onto Overlay dialog box opens.
3. Select the shapefile you want to import.
4. Click Open.



47. Creating and Editing Tactical Graphics and Control Objects

This chapter explains how to create and edit tactical graphics.

47.1. Creating Tactical Graphics 1012
47.1.1 How Areas Are Drawn 1012
47.1.2 Creating Freehand Lines 1012
47.1.3 Drawing Circles and Ellipses 1013
47.1.4 Drawing Arcs 1014
47.1.5 Drawing Sector Areas 1014
47.1.6 Drawing Spider Webs 1015
47.1.7 Drawing Boxes 1016
47.1.8 Creating Volumetric (Extruded) Tactical Graphics 1016
47.1.9 Adding Text to an Overlay 1017
47.1.10 Assigning Tactical Graphics to an Overlay 1017
47.1.11 Displaying a Terrain Profile When You Create a Line 1018
47.1.12 Publishing Tactical Graphics 1019
47.2. Creating a Computed Route 1019
47.3. Creating an Intel Collection Area 1020
47.4. Creating and Editing a Minefield Area 1021
47.5. Creating Ditches, Berms, and Tank Scrapes 1023
47.6. Creating Nav Edges 1023
47.7. Creating and Using Bullseyes 1025
47.7.1 Measuring Bearing and Range 1026
47.7.2 Attaching a Spider Web to a Bullseye 1028
47.8. Attaching Tactical Graphics to a Simulation Object 1028
47.8.1 Attaching a Tactical Graphic Automatically 1029
47.9. Deleting Tactical Graphics 1029
47.10. Editing Tactical Graphics 1029
47.10.1 Editing Vertices 1030
47.10.2 Resizing Boxes and Text Objects 1032

47.10.3 Rotating an Ellipse 1032
47.10.4 Changing the Direction of a Line 1033
47.10.5 Changing the Color of a Tactical Graphic 1033
47.10.6 Changing Linear and Areal Style Properties 1033
47.11. Active and Inactive Tactical Graphics 1035

47.1. Creating Tactical Graphics

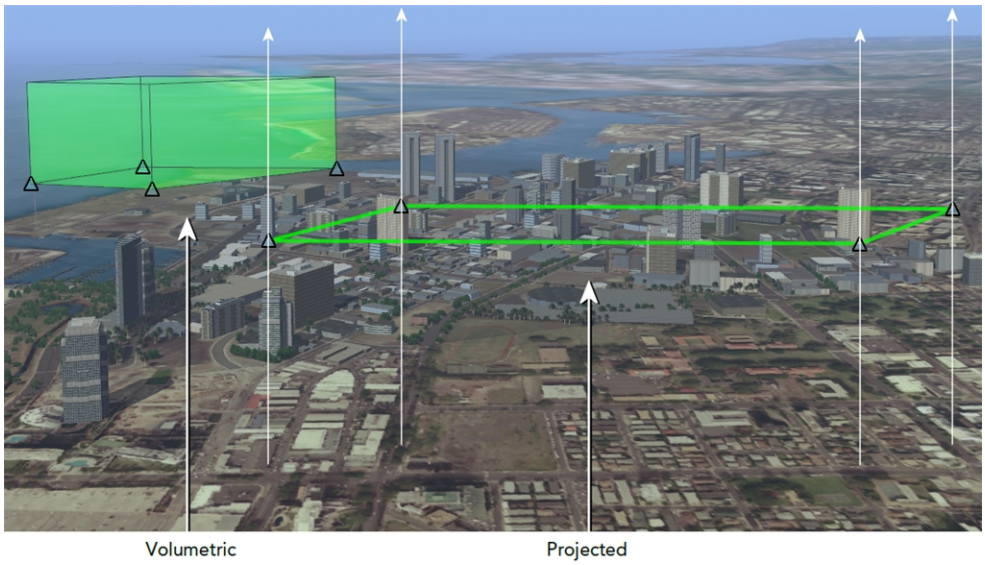
The generic procedures for creating tactical graphics are described in "Chapter 14. Creating and Placing Objects" on page 371. This section provides additional details about tactical graphic creation and placement.

47.1.1 How Areas Are Drawn

VR-Forces supports two types of areas, projected and volumetric (Figure 352). The points in a projected area do not have an altitude. They define an area that projects from ground level to an infinite height. Volumetric areas have a floor and a ceiling.

When you create a projected area, the points are drawn using the altitude of the first point in the area. This may result in some or all of the boundary of the area being drawn in the air. This is simply a matter of how it is drawn. The area still is projected from ground level upwards.

Figure 352. Volumetric and projected areas



47.1.2 Creating Freehand Lines

A freehand line is essentially many short line segments. After you create a freehand line, you can change the sampling rate used to determine how many segments it has. The fewer the segments, the less fluid the line is.

Unlike other lines, the individual vertices of a freehand line do not have vertex markers. You cannot select them individually on the map. However, you can edit them in the Edit Freehand Line dialog box.

The Freehand Line With Arrow tactical graphic automatically puts an arrow at the end of the line. However, you can also add an arrow to a freehand line that is created without one.

> If you reduce the sampling rate, resulting in fewer segments and a less fluid line, once you save the line you cannot automatically increase the number of segments to increase the fluidity of the line. You would have to add vertices by hand.

➤ **To create a freehand line:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select Freehand line.
4. Click where you want the line to start and, while holding down the left mouse button, draw a line on the screen.
5. Release the mouse button. The Create Freehand dialog box opens.
6. Edit the properties as desired.

47.1.3 Drawing Circles and Ellipses

Simulation objects do not interact with circles and ellipses. They are strictly for use as graphics.

> If you have cleared the Attach Object to Mouse check box in the Create Circle or Create Ellipse dialog box, you must still define the center point and other dimensions before the Create button is enabled. Since the object is not attached to the mouse, you will not see dynamic resizing of the object as you specify the points.

➤ **To draw a circle:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select Circle. The cursor changes to draw mode and shows a vertex icon.
4. Click to place the center of the circle. A center point is placed on the terrain and a circle is drawn. There is now a vertex icon on the circumference of the circle.
5. Drag the mouse to set the radius of the circle.
6. Click to complete the circle.

➤ **To draw an ellipse:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ( ).
3. Select Ellipse. The cursor changes to draw mode and shows a vertex icon.
4. Click to set the center of the ellipse. A center point is set and a small circle is displayed with a vertex indicator at the bottom of the circle.
5. Drag the mouse up or down and click to set the height of the ellipse. A vertex indicator is added to the left side of the circle.
6. Drag the mouse left and right and click to set the width of the ellipse.

47.1.4 Drawing Arcs

Simulation objects do not interact with arcs. They are strictly for use as graphics.

➤ **To draw an arc:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ( ).
3. Select Arc. The cursor changes to draw mode and shows a vertex icon.
4. Click to place one end of the arc.
5. Click to place the other end of the arc.
6. Drag the mouse to set the radius of the arc.
7. Click to complete the arc.

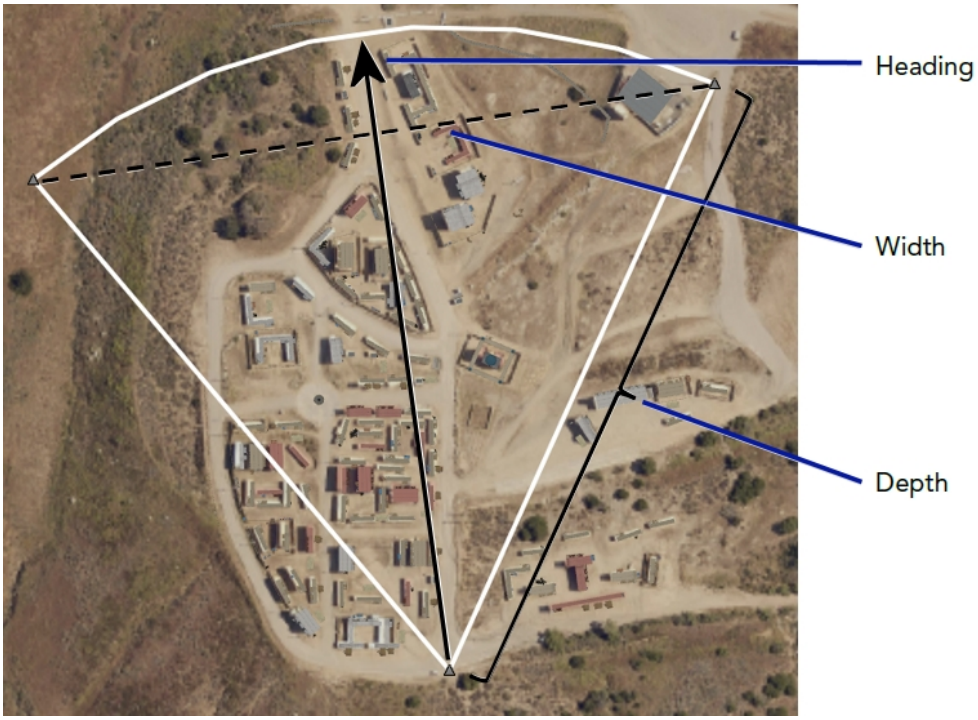
47.1.5 Drawing Sector Areas

Sector areas are defined by two legs with a common origin whose endpoints are connected by an arc (Figure 353). Drawn clockwise, the sector can be up to 360 degrees (a full circle).

In addition to the attributes typical to tactical graphics, you can manually specify these attributes in the Edit Sector dialog box:

- Width. The length of a line drawn between the end points of the two legs of the sector.
- Depth. The length of a line from the origin to the center of a line drawn between the end points of the two legs of the sector.
- Heading. The heading of a line drawn from the origin to the center of the arc of the sector.

Figure 353. Sector area



➤ **To draw a sector area:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ( ).
3. Select Sector. The cursor changes to draw mode and shows a vertex icon.
4. Click to place the common vertex of the sector.
5. Move the mouse and click to locate the end of one leg of the sector. (Do not hold down the mouse button while you move it.)
6. Move the mouse and click to locate the end of the other leg of the sector.

47.1.6 Drawing Spider Webs

Spider webs display sectors and ranges. Its heading is oriented to either true or magnetic north, based on the Display Units Settings. Each spoke displays its heading. In addition to the attributes typical to tactical graphics, you can manually specify these attributes in the Create Spider Web dialog box:

- Number of Rings. The number of rings to include in the spider web.
- Distance Between Rings (NM). The distance between each ring in the spider web in nautical miles.

  The spider web sizes itself based on the number of rings and the distance between them that you specify.
- Number of Spokes. The number of spokes in the spider web.

➤ **To draw a spider web:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select Spider Web. The cursor changes to draw mode and shows a vertex icon.
4. Open the Create Spider Web dialog box and then edit the parameters as desired.
5. Click where you want to place the center of the spider web. It is created using the parameters you specified.

47.1.7 Drawing Boxes

Boxes are constrained to be parallelograms. Simulation objects do not interact with boxes. (In other words, they are not areas.)

➤ **To draw a box:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select Box. A box is attached to the cursor
4. Click to place the box in the approximate location that you want.
5. Double-click any vertex. The vertex changes to show it is editable.
6. Drag the vertex to reshape the box.

47.1.8 Creating Volumetric (Extruded) Tactical Graphics

You can create volumetric tactical graphics. Volumetric tactical graphics are useful for defining air corridors, airspace, and similar 3D areas. Some volumetric graphics you can create are:

- Area (extruded).
- Box (extruded).
- Circle (extruded).
- Corridor (a linear object).
- Ellipse (extruded)
- Joint Engagement Zone (Altitude Limited)
- Missile Engagement Zone (Altitude Limited).

Volumetric areal objects have a floor and a ceiling above the terrain. Corridors have a width and height, which are centered on the vertices. Simulation objects can move along a corridor. (They move along the center line and ignore the height and width.)

The Entity in Area condition checks whether an entity is within the volume of an extruded area.

➤ **To create a volumetric tactical graphic:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select the volumetric tactical graphic you want to create.
4. Specify the vertices of the tactical graphic just like you would for a 2D tactical graphic. When you right-click to set the final vertex, the Edit dialog box should open.
5. If the Edit dialog box does not open, click its tab to open it.
6. If you are creating a route, specify the height and width. If you are creating any other extruded tactical graphic, specify the floor and ceiling (above mean sea level).
7. Click Create.

47.1.9 Adding Text to an Overlay

You can add text to an overlay. The maximum width of the text object is fixed.

➤ **To add text to an overlay:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ().
3. Select Text. The text "Text" is attached to the cursor.
4. Click to set the bottom center of the text. The Create Text dialog box opens.
5. Type the text you want in the Text box.
6. Edit any other properties as desired.
7. Click Create.

47.1.10 Assigning Tactical Graphics to an Overlay

By default, when you create a tactical graphic, it is assigned to the Tactical Graphics overlay. If no overlay exists, the overlay is created and the graphic is assigned to it. You can assign a tactical graphic to a different overlay when you create it or edit it. To assign a tactical graphic to an overlay when you create it, set this property as described in "14.3.2 Specifying an Object's Properties Before You Create It (Click to Locate)" on page 384. To move a tactical graphic to a different overlay, follow the procedure in "Moving a Tactical Graphic to a Different Overlay" below.

Moving a Tactical Graphic to a Different Overlay

➤ **To move a tactical graphic to a different overlay:**

1. Select the tactical graphic you want to move.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. In the Overlay list, select the overlay that you want the tactical graphic to move to.
5. Click Update.

> You can also move a tactical graphic to a different overlay by dragging it to the new overlay on the Overlays View of the Objects List Panel.

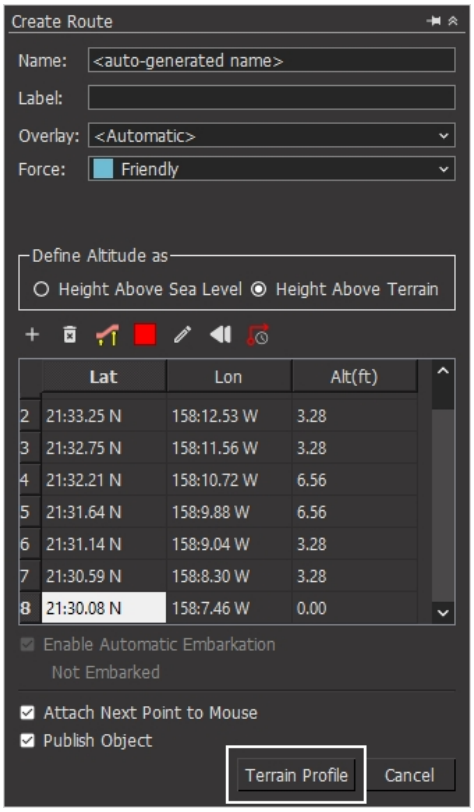
47.1.11 Displaying a Terrain Profile When You Create a Line

You can display a terrain profile window while you create a line. You can configure VR-Forces to automatically display a terrain profile when a line is created "65.1.2 Configuring Terrain Profiles" on page 1267), or you can display the window manually.

➤ **To open a terrain profile window while you are creating a line:**

1. Select the tactical graphic that you want to create as described in "14.2. Selecting the Object to Create" on page 373. A Create *object* tab is added to the window below the palettes).
2. Click the Create *object* tab. A Create *object* dialog box that is appropriate for the object opens (Figure 354).

Figure 354. Terrain Profile button



3. Click to place the first vertex. The Terrain Profile button is activated.
4. Click Terrain Profile. A terrain profile window opens.
5. Complete the line. The terrain profile window stays open until you complete the line.

47.1.12 Publishing Tactical Graphics

By default, all tactical graphics are published. That is, they are sent out over the network and are visible to other participants in an exercise. You can choose to not publish a tactical graphic. In that case, it is visible only in the GUI on which it was created.

Published tactical graphics save to the scenario objects file. Unpublished tactical graphics save to the overlays file that is created as part of the scenario. You can also manually save tactical graphics to an overlay file separate from the one stored with the scenario.

➤ **To publish or unpublish a tactical graphic:**

1. Select the graphic you want to edit.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. To publish the object, select the Publish Object check box. To unpublish it, clear the check box.

47.2. Creating a Computed Route

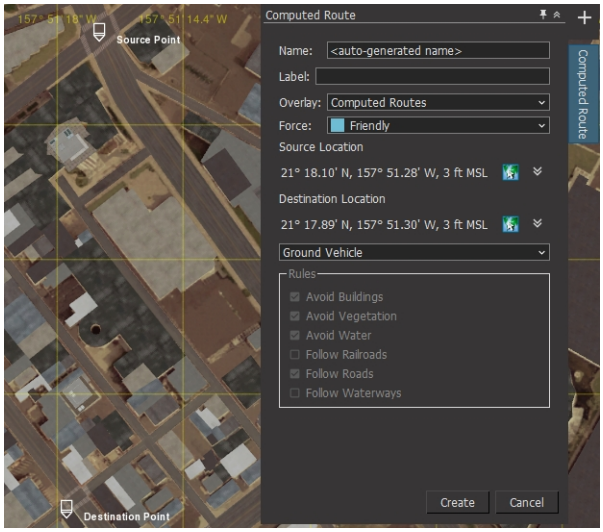
VR-Forces can create routes between two points and take into account criteria such as using roads and avoiding buildings and water. Unlike the temporary routes that VR-Forces creates for movement tasks, a computed route is a tactical graphic you save with the scenario and any simulation object can use.

When you create a computed route, you specify rules that VR-Forces should use when creating the route. VR-Forces includes preconfigured rule sets for ships and ground vehicles. You can add additional preconfigured rule sets. You can also add additional rules. The rules and rule sets are configured in *./data/simulationModelSets/<sms>/gui/computedRouteRulesTypes.xml*.

➤ **To create a computed route:**

1. Choose **Create > Computed Route**. The Create Computed Route dialog box opens (Figure 355).

Figure 355. Creating a computed route



2. Optionally, specify a name, label, and overlay.
3. Select the force for the route.
4. Click to place the starting point of the route, or expand the location group box by clicking the arrow next to enter a value for the Source Location.
5. Click to place the end point of the route, or enter a value for the Destination Location.
6. Select the type of vehicle from the list. Ground vehicles and ships have a preconfigured set of rules. Select Custom to choose a different set of rules.
7. If you select Custom, select the rules that you want to enforce for the route. Clear those you do not want to use.
8. Click Create.

47.3. Creating an Intel Collection Area

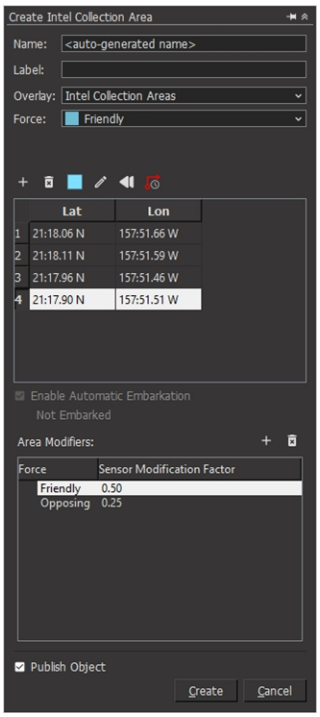
Intel collection areas modify the sensor signatures of simulation objects within their boundaries. This affects the ability of other simulation objects to detect them. Intel Collection Areas are created just like other areas. The difference is specification of area modifiers. Area modifiers are set per force. A modifier greater than 1.0 increases the sensor signature for members of that force, making them easier to detect. A modifier less than 1.0 reduces their signature, making them harder to detect.

➤ **To create an intel collection area:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ( ).
3. Select Intel Collection Area.
4. Draw an area on the map as you would for any other tactical graphic area.

5. In the Create Intel Collection Area dialog box (Figure 356), click the Add button (+) next to the Area Modifiers label. The Choose Force dialog box opens.

Figure 356. Create Intel Collection Area dialog box



6. Select the force for which you want to add a modifier.
7. Click OK. The force is listed in the Area Modifiers window.
8. Double-click the Modifier column to make it editable.
9. Type the modifier value you want to apply to this force.
10. Optionally, add additional force entries to the list.
11. Click Create.

47.4. Creating and Editing a Minefield Area

There are two types of tactical graphics for minefield areas: tactical graphics that simply represent areas where minefields could be in the scenario but have no effect, and simulation effect objects that create minefields with the parameters that you specify. The latter can damage entities that move through the area.

The tactical graphic minefield areas are:

- Minefield Antipersonnel. Represents a minefield designed for infantry.
- Minefield Antitank, Minefield Antitank Antihandling, Minefield Antitank Wide Area. Represent minefields designed for tanks.
- Minefield Decoy, Minefield Decoy Fenced. Represent decoy minefields.
- Minefield Graphic, Minefield Unspecified. Represent minefields of no specific type.

- Planned Minefield. Represents where a minefield could be placed in the future.
- Unexploded Explosive Ordinance. Represents ordinance that has not yet exploded.

The simulation effect object minefield areas are:

- Minefield Conventional. A minefield that can include conventional antitank and antipersonnel mines, which you define one you draw the minefield area. Conventional mines cannot self-destruct.
- Minefield Scatterable. A minefield that can include remote anti-armor and area denial anti-personnel mines, which you define once you draw the minefield area. Scatterable mines can self-destruct.
- Minefield RAAM-ADAM. A scatterable minefield that includes two types of mines by default—M731 scatterable and M73 RAAM-ADAM mines—which are configured to self-destruct after 48 hours pass in the simulation. You can edit the minefield after you draw it and change the mines in the minefield.

The Minefield Conventional and Minefield Scatterable objects require you to specify what mines are in the minefield when you create the area. Until you do, they will not damage entities that move into those areas. Minefield RAAM-ADAM comes with predefined mines and therefore is primed and ready to go when you place it.

➤ **To create a minefield:**

1. Open the Create Objects Palette.
2. Optionally, select the filter for Tactical Graphic () or Simulation Effect Object ().
3. In the objects list, select the minefield you want to create.
4. Draw an area on the map as you would for any other tactical graphic or control object area.
5. If you are creating a minefield tactical graphic or a RAAM-ADAM minefield object, the area displays on the map. Tactical graphics require no more work.

   If you are creating a conventional or scatterable object, the Create Minefield panel opens. If you are creating a RAAM-ADAM minefield object, you can edit the minefield using that panel. Specify the type of mines for the minefield:

   a. To add a mine type, click the Add button () in the lower area. The Configure Mine dialog box opens.
   b. From the Munition Type list, select the type of mine to add to the minefield.
   c. From the Fuse Type list, select the type of fuse for the mine.
   d. If you specified a trip wire fuse, specify the length for the trip wire.
   e. Click OK.
   f. In the Create Minefield panel, specify the areal density for the mine type, which is the number of mines per unit of measurement.

g. If you are creating a scatterable minefield, you can specify the number of hours before the mine self destructs.

h. Add more mine types to the minefield as needed.

i. Click Create. The minefield displays on the map.

47.5. Creating Ditches, Berms, and Tank Scrapes

Ditches, berms, and tank scrapes are dynamic terrain objects that you can place to change the ground in parts of osgEarth terrains defined by `heightfield` elements.

- These dynamic terrain objects do not work in most high fidelity inset areas.
- The dynamic terrain objects are configured in the *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef* file by the `linear-deformation` parameters in the `linear-deformation-list` block. Tank scrapes are directional, providing a place for tanks to take hull-down positions. The elevation deformation of tank scrapes is in *./SharedData/19/latest/ModelData/Environment/Land/TankScrape*.

➤ **To create a dynamic terrain object:**

1. Open the Create Objects Palette.
2. Optionally, select the filter for Terrain ().
3. In the objects list, select the dynamic terrain object you want to create.
4. Draw a line as you would for any other linear object.
   - Ditches and berms can have two or more points.
   - Tank scrapes are a two-point directional line.

   After you finish drawing the line, the terrain deforms to conform to the line you drew.

➤ To restore the terrain, simply delete the object.

47.6. Creating Nav Edges

Navigation edges are tactical graphics—specifically terrain simulation effect objects—that affect the navigation mesh in a navigation area. They permit VR-Forces to calculate paths to locations that an entity would otherwise not be able to move to.

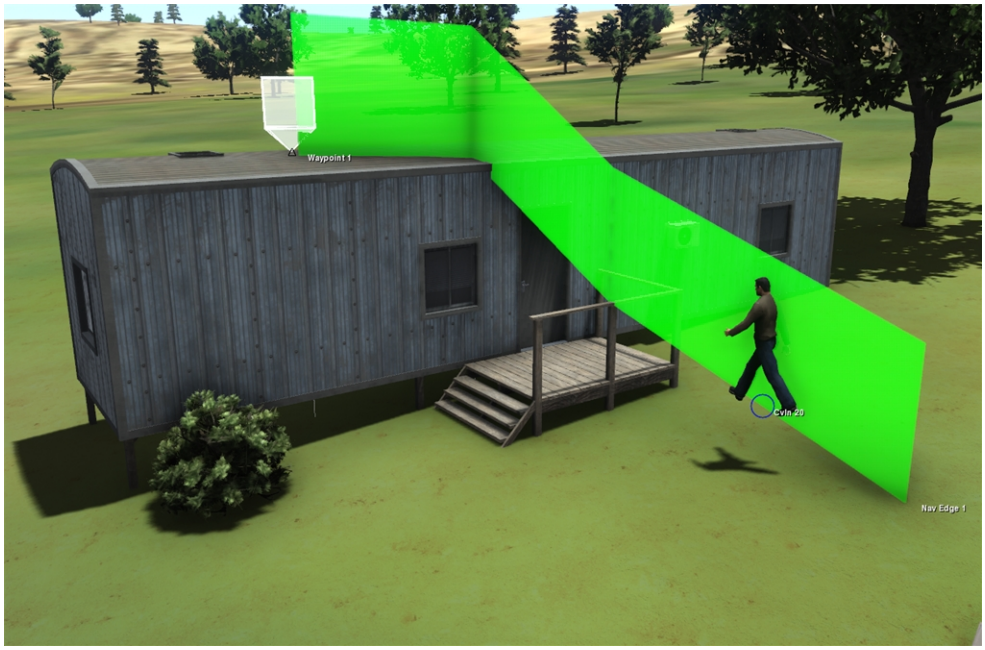
For example, suppose that you want an entity to walk from the ground to a waypoint on top of a building that does not have any interior geometry. Normally, the task would fail with a message, "No path to destination" as in Figure 357

Figure 357. Pedestrian cannot move to waypoint



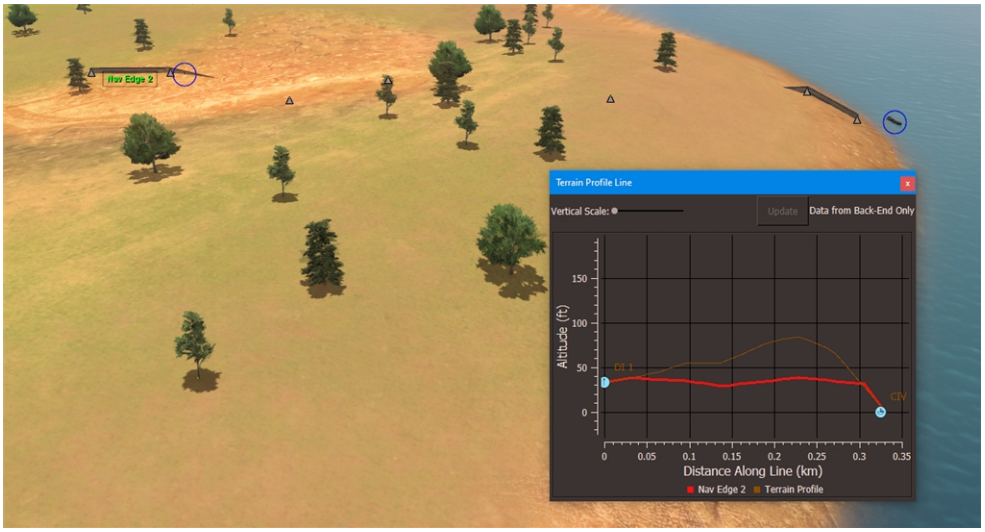
By adding a Direct Movement Nav Edge simulation effect object, the entity can plan a path to the waypoint (Figure 358). Use of nav edges in this way may compromise the realism of entity movement, but may be acceptable to achieve the overall goals of a simulation if the terrain is not as detailed as you would like it to be.

Figure 358. Navigation edge allows path planning



Another use for nav edges is to draw them with vertices that are underground, effectively creating tunnels that do not exist in the created terrain (Figure 359).

Figure 359. Entity follows nav edge underground



Finally, you might use a nav edge in cases where navigation data exists, but it is too restrictive. For example, the bounding volume of an entity might be slightly too large to permit it to go through a door. Adding a nav edge would allow it to go through the door.

VR-Forces has these types of nav edge simulation effect objects:

- Direct Movement Nav Edge. A generic nav edge.
- Pedestrian Nav Edge. A nav edge that acts like a pedestrian path. Pedestrians will prefer to use it over generic paths.
- Crawl Nav Edge. When a human that has a crawl animation moves along this route, it crawls.

You can also add nav edges in a navigation profile. For details, see "66.3.1 Navigation Profiles" on page 1281.

➤ **To create a nav edge:**

1. In the Create Object Palette, select a nav edge. (Selecting the Terrain simulation effect filter ( ) and entering "nav" in the search box quickly lists the available types of navigation simulation effect objects.)
2. Draw the nav edge as you would a linear tactical graphic.

47.7. Creating and Using Bullseyes

You can place a bullseye in the scenario and then use that to check the bearing and range between the bullseye and a simulation object or the mouse cursor. You create bullseyes in the same manner as other tactical graphics, as described in "Chapter 14. Creating and Placing Objects" on page 371.

By default, bullseyes are added to the bullseye overlay. You can assign them to another one; see "47.1.10 Assigning Tactical Graphics to an Overlay" on page 1017.

➤ **To create a bullseye:**

1. Open the Create Object Palette.
2. Optionally, click the Tactical Graphics filter ( ).
3. Select Bullseye.
4. Place the bullseye on the terrain.

➤ To open the Bullseye Selection Toolbar, choose **View > Toolbars > Bullseye Selection Toolbar**.

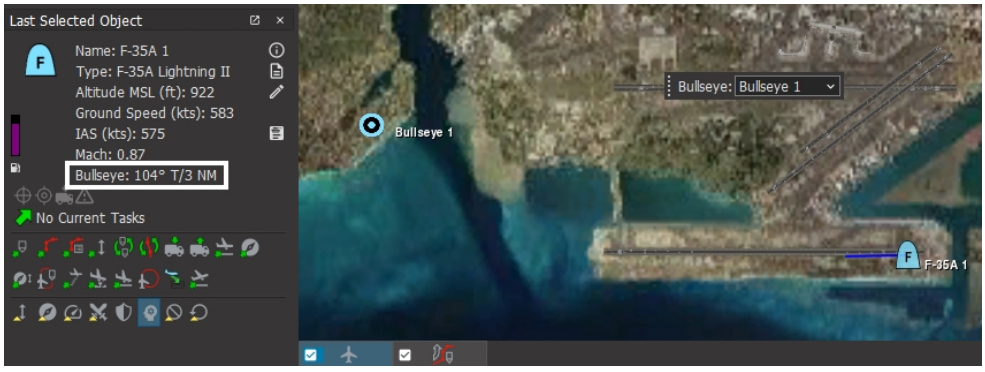
47.7.1 Measuring Bearing and Range

You can check the bearing and range between the bullseye and a simulation object or the cursor. Using the cursor, you can only check the bearing and range in the Plan View. The Last Selected Object panel simulation objects always displays the bearing and range from a bullseye, regardless of the view.

➤ **To check the bearing and range from a bullseye to a simulation object:**

1. By default, the first (or only) bullseye in the scenario is selected. If you want to measure the distance to another bullseye, select the bullseye from the list in the Bullseye Selection Toolbar.
2. Select the simulation object. The bearing and range from the selected bullseye displays in the Last Selected Object panel.

Figure 360. Bearing and range from bullseye to object



➤ **To display the bearing and range from a bullseye to the cursor at the bottom:**

1. In the Bullseye Selection Toolbar, select the bullseye you want from the list.
2. Choose **Observer > Mouse Bullseye Position**. The From Bullseye information displays at the bottom of the window.
3. Move the cursor around the terrain. The bearing and range from the bullseye update based on the location of the cursor.

Figure 361. Bearing and range from bullseye to cursor (bottom)



➤ **To display the bearing and range from a bullseye beside the cursor:**

1. In the Bullseye Selection Toolbar, select the bullseye you want from the list.
2. Choose **Observer > Map Cursor Position From Bullseye On Cursor**. The bearing and range display next to the cursor.
3. Move the cursor around the terrain. The bearing and range from the bullseye update based on the location of the cursor.

Figure 362. Bearing and range from bullseye to cursor (beside cursor)



You can change the font and color of the information attached to the cursor by editing the DefaultMapMouseCursorFromBullseyeOnMouseOverlay model definition. To learn about editing model definitions, see "Creating and Editing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.

47.7.2 Attaching a Spider Web to a Bullseye

You can set a spider web on a bullseye. Spider webs display sectors and ranges. The spider webs you can attach to a bullseye come in three sizes, measured in nautical miles (NM).

➤ **To attach a spider web to a bullseye:**

1. Select the bullseye.
2. Choose **Set** and then the size of spider web you want to use:
   - **Display 70 NM Range Rings**
   - **Display 100 NM Range Rings**
   - **Display 200 NM Range Rings**

➤ To remove the spider web from a bullseye, select the bullseye and then choose **Set > Clear Range Rings**.

47.8. Attaching Tactical Graphics to a Simulation Object

You can attach tactical graphics to a simulation object so that they move with it. For example, if you draw a route on an aircraft carrier deck, you want the route to stay attached to the deck as the carrier moves. Another example is to attach a Terrain Page-in Area to an aircraft so that it pages in terrain as the aircraft flies.

You can attach a tactical graphic to an object manually or automatically (3D observer modes only).

➤ **To attach a tactical graphic to a simulation object manually:**

1. Create a tactical graphic as you normally would.
2. Select the tactical graphic.
3. Choose **Objects > Attach Object To**. The Attach Object To dialog box opens.
4. Select the simulation object to which you want to attach the tactical graphic.
5. Click OK.



You can attach tactical graphics to entities using the Attach set data request, as described in "40.12. Attached" on page 871.

47.8.1  Attaching a Tactical Graphic Automatically

You can attach a tactical graphic to a 3D model as you create it. (This procedure uses the automatic embarkation capability. For information about embarkation, see "Chapter 15. Embarked and Embedded Entities" on page 405.)

➤ **To attach a tactical graphic to a 3D model when you create it:**

1. Select a 3D observer mode, such as Stealth mode.
2. Move the observer to view the simulation object on which you want to attach a tactical graphic.
3. On the Create Object Palette, select the tactical graphic that you want to create.
4. Move the cursor over the 3D model you want to attach to until you see a green box around the model.
5. Click to place the first vertex (or point for point tactical graphics) on the 3D model.
6. Continue to create the tactical graphic as you normally would. The first point remains anchored to the model.

47.9. Deleting Tactical Graphics

➤ **To delete a graphical object:**

1. Select the tactical graphic you want to delete.
2. Choose **Objects > Delete**, or press the **Delete** key.

47.10. Editing Tactical Graphics

You can edit a tactical graphic's properties. The properties that you can edit vary depending on the type of graphic and can include some or all of the following:

- Add, delete, and edit vertices ("49.1. Displaying Remote Graphics" on page 1044).
- Change its name.
- Change the altitude reference ("14.4.2 Setting Altitude in the Create Object or Edit Object Dialog Box" on page 387, "14.4.3 Specifying the Altitude for All of the Vertices in a Route" on page 388).
- Change the direction of a line ("47.10.4 Changing the Direction of a Line" on page 1033).
- Edit linear and areal styles:
  - Change the pen style and width.
  - Change the line style.
  - Add or remove arrows.
  - Change the fill of an area.

  ("47.10.6 Changing Linear and Areal Style Properties" on page 1033.)

- Move it to a different location ("14.9.1 Dragging an Object to a New Location" on page 392).
- Move it to a different overlay ("Moving a Tactical Graphic to a Different Overlay" on page 1017).
- Change the color ("47.10.5 Changing the Color of a Tactical Graphic" on page 1033).
- Resize boxes and text objects ("47.10.2 Resizing Boxes and Text Objects" on page 1032).
- Rotate an ellipse ("47.10.3 Rotating an Ellipse" on page 1032).

To edit a tactical graphic, its overlay must be unlocked.

47.10.1 Editing Vertices

You can edit the vertices of routes and areas. You can edit them dynamically or in the Edit Object dialog box.

Adding a Vertex

You can add vertices to a route or area directly on the terrain or in the Edit Object dialog box.

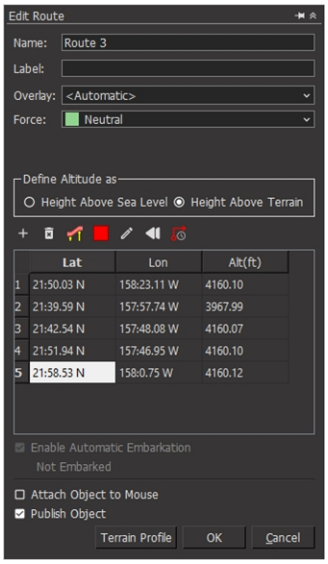
➤ **To add a vertex to a tactical graphic dynamically:**

1. Right-click the vertex next to which you want to add a new vertex. A menu displays.
2. Choose **Add Point After** or **Add Point Before**. A point is added midway between the selected vertex and the next or previous vertex, based on the command you chose.
3. Optionally, edit the new point as described in "Editing a Vertex" on the facing page.

➤ **To add a vertex in the Edit Object dialog box:**

1. Select the tactical graphic you want to edit.
2. Choose **Objects > Edit**. The Edit Object dialog box opens (Figure 363).

Figure 363. Edit Route dialog box



3. If Attach Object to Mouse is selected, clear the check box.
4. In the list of vertices, select the vertex after which you want to add the new vertex.
5. Click the Add button (+). A vertex is added midway between the selected vertex and the next vertex. If you selected the last vertex, a new vertex is added to the end of the route.
6. Optionally, edit the new point as described in "Editing a Vertex" below.
7. Click Update.

Editing a Vertex

You can edit a vertex dynamically or in the Edit Object dialog box.

➤ **To edit a vertex dynamically:**

1. Double-click the vertex.
2. Move the mouse to change the vertex's location.
3. Use **Alt**+mouse wheel or **Alt**+left mouse button+drag to change the altitude.
4. Click to place the vertex.

➤ **To edit a vertex in the Edit Object dialog box:**

1. Select the tactical graphic you want to edit.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. If you are editing a waypoint, click on the terrain to specify a new location or edit the coordinates and altitude in the Edit Waypoint dialog box.

   If you are editing a phase line, route, or area, you can:

a. Double-click the coordinate value you want to change to make it editable.

b. Type a new value and press **Enter** or click someplace outside of the value.

Alternatively, you can click on the map to change the coordinates of the selected vertex. To set the altitude, you must edit the value by hand.

5. Click Update.

Deleting a Vertex

You can delete vertices directly or in the Edit Object dialog box.

➤ **To delete a vertex directly:**

1. Right-click the vertex you want to delete. A menu displays.
2. Choose **Delete Point**. The point is deleted. (There is no confirmation prompt.)

➤ **To delete a vertex in the Edit Object dialog box:**

1. Select the tactical graphic you want to edit.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. If Attach Object to Mouse is selected, clear the check box.
4. In the list of vertices, select the vertex you want to delete.
5. Click the Remove Point button ( ).
6. Click Update.

47.10.2 Resizing Boxes and Text Objects

You cannot edit the individual vertices of a box or a text object, you can only resize them.

➤ To resize a box or text object, double-click one of its vertices to make it editable and drag it to create the new object size.

47.10.3 Rotating an Ellipse

In addition to changing the width and height of an ellipse, you can rotate it around its center point. You can undo and redo ellipse rotation.

➤ **To rotate an ellipse:**

1. Select the ellipse.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. Select the Free Rotate check box.
4. Click OK.
5. Double-click one of the points on the ellipse to make it editable and drag it in a circular direction.

47.10.4 Changing the Direction of a Line

All lines have a direction. The direction affects how a simulation object moves along it and the direction that arrows face, if present. If you draw a line and then decide you want to reverse its direction, you can do so.

➤ **To change the direction of a line:**

1. Select the line.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. Click the Reverse Direction button ( ).
4. Click OK.

47.10.5 Changing the Color of a Tactical Graphic

By default, most tactical graphics are colored for the specified force. You can change the color of a tactical graphic.

➤ **To change the color of an tactical graphic:**

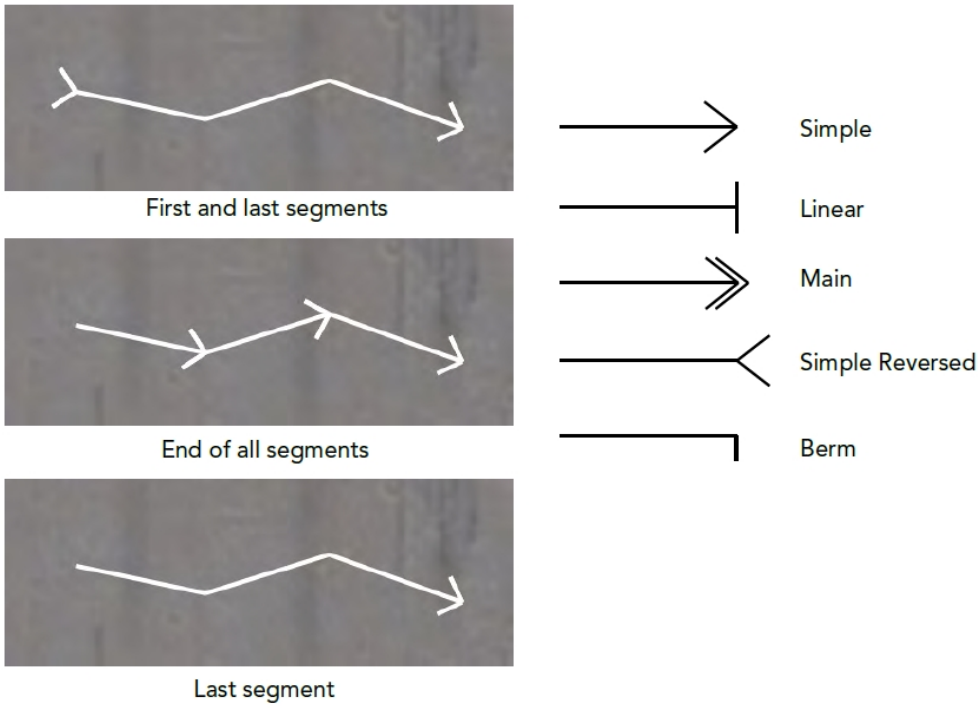
1. Select the graphic whose color you want to change.
2. Choose **Objects > Edit**. The Edit Object dialog box opens.
3. Click the Color button ( ). A color picker dialog box opens.
4. Click a color button or click the Custom Color button to choose a new color from the Color Picker.
5. In the Color Picker, click OK.
6. In the Edit Object dialog box, click OK.

47.10.6 Changing Linear and Areal Style Properties

Lines and areas have a variety of properties that you can set in the Edit Style dialog box. The procedure is essentially the same for all of these properties. The properties you might be able to change are:

- Line width. The width of the lines that make up a tactical graphic, either the actual line or the boundary of an area.
- Pen style. How the line or outline is drawn – solid line, no line, or as dashed and dotted lines. This property does not apply to lines that use a texture, such as Line of Contact.
- Fill style. The fill style for areal objects. Fill styles include degrees of solid fill and various types of line and grid fills.
- Line style. The different types of special lines, such as Main Axis.
- Arrow. You can add arrows to the end of a line, at the end and beginning segment, or at the end of each segment of the line. Figure 364 illustrates the arrow styles.

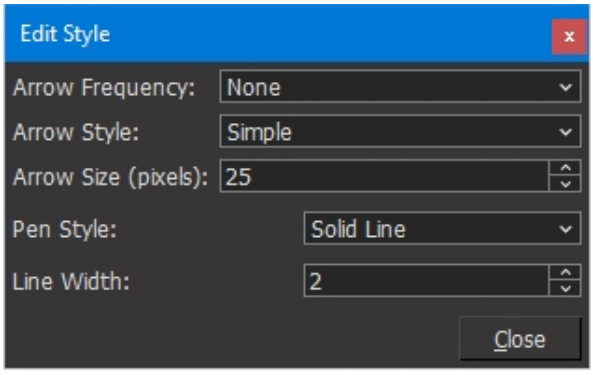
Figure 364. Arrow styles



➤ **To change a style property:**

1. Select the graphic you want to edit.
2. Choose **Objects > Edit**. The Edit object dialog box opens.
3. Click the Edit Style button (). The Edit Style dialog box opens (Figure 365). It lists the style options that you can edit for this tactical graphic.

Figure 365. Edit Style dialog box



4. Change the properties by selecting options from the appropriate list or typing new values.
5. Click OK.

47.11. Active and Inactive Tactical Graphics

You can activate (display) and deactivate (hide) all published tactical graphics. You can set a tactical graphic's state when you create it or edit it, and using a set data request. You can test for a tactical graphic's state with the Tactical Graphic Active condition. You can set a tactical graphic's state based on simulation time or time of day.

> For inactive tactical graphics to be hidden, you must select Show Only Active Environmentals on the Display Settings dialog box, Tactical Graphics Settings page or click the Show Only Active Tactical Graphics button ( )on the Display Settings Toolbar.

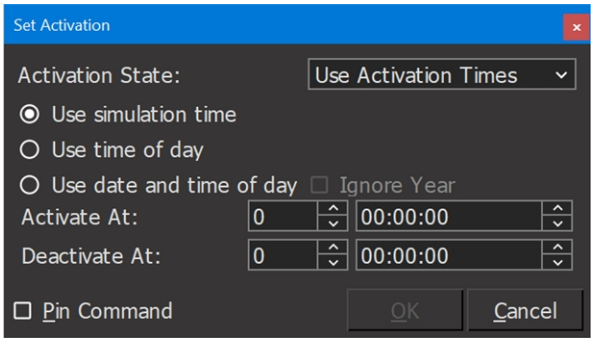
In most cases, deactivating a tactical graphic is a GUI effect. The graphic continues to be simulated and published over the network. Simulation objects continue to interact with it. However, if you set an indirect fire event area to be inactive, and you specify that the detonation start time is Wait for Activation, the indirect fire event does not begin until the area is activated.

For details about the Activation set data request, see "48.2. Activation" on page 1038 and "Tactical Graphic Active" on page 942. For details about indirect fire events, see "10.1. Introduction to Indirect Fire" on page 314.

➤ **To set the activation state of a tactical graphic:**

1. Select the tactical graphic you want to edit. (If you are creating a new tactical graphic, open the Create object dialog box and skip to step 3.)
2. Choose **Objects > Edit**. The Edit object dialog box opens.
3. Click the Set Activation button ( ). The Set Activation dialog box opens (Figure 366).

Figure 366. Set Activation



4. Select one of these options from the Activation State list:
   - Active. The tactical graphic is visible.
   - Inactive. The tactical graphic is not visible if Show Only Active Environmentals is selected on the Display Settings dialog box, Tactical Graphics Display Settings page.
   - Use Activation Times. The tactical graphic is active during the times specified in the dialog box.
5. If you selected Active or Inactive, click OK. If you selected Use Activation Times:
   a. Select Use Simulation Time or Use Time of Day.
   b. In the Activate At boxes, specify the time you want the tactical graphic to be visible.
   c. In the Deactivate At boxes, specify the time you want the tactical graphic to be hidden.
6. Click OK.



48. Setting the State of Tactical Graphics

You can use set data requests to change the state of a tactical graphic.

48.1. Set Data Requests for Tactical Graphics 1038
48.2. Activation 1038
48.3. Allow Crossing 1039
48.4. Arrowhead Style 1039
48.5. Color 1040
48.6. Fill Style 1040
48.7. Force 1040
48.8. Line Width 1040
48.9. Notify Level 1041
48.10. Pen Style 1041

48.1. Set Data Requests for Tactical Graphics

Set data requests for tactical graphics provide an alternative to using Edit dialog boxes to change the state of a graphic. The advantage of using set data requests is that you can make changes to tactical graphics from plans and in scripted tasks or scripted sets. You can also quickly make changes from the Set menu without needing to open an Edit dialog box.

Aerodrome and runway set data requests are covered in "67.3. Aerodrome and Runway Set Data Requests" on page 1295.

48.2. Activation

Use the Activation set data request to activate or deactivate tactical graphics. In most cases, this is strictly a visual effect; activated tactical graphics are visible; inactive tactical graphics are not visible. However, in the following cases, setting a tactical graphic to be inactive affects how it functions:

- If you set an indirect fire event area to be inactive, the indirect fire event does not start until the area is activated.
- If you set a spawn point to be inactive, it does not spawn entities.

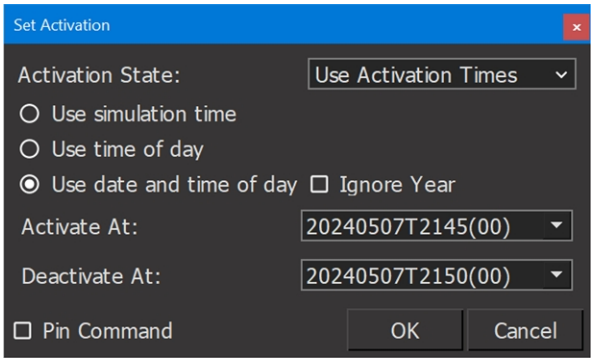
Inactive tactical graphics continue to be published and simulation objects interact with them normally. You can use the Tactical Graphic Active condition to test the state of a tactical graphic.

For more information, see "47.11. Active and Inactive Tactical Graphics" on page 1035 and "Tactical Graphic Active" on page 942.

➤ **To activate or deactivate a tactical graphic:**

1. Select the tactical graphic.
2. Choose **Set > Disposition > Activation**. The Activation dialog box opens (Figure 367).

Figure 367. Activation



3. Select one of these options from the Activation State list:
   - Active. The tactical graphic is visible.
   - Inactive. The tactical graphic is not visible if Show Only Active Environmentals is selected on the Display Settings dialog box, Tactical Graphics Display Settings page.
   - Use Activation Times. The tactical graphic is active during the times specified in the dialog box.
4. If you selected Active or Inactive, click OK. If you selected Use Activation Times:
   a. Select Use Simulation Time, Use Time of Day, or Use Date and Time of Day.
   b. In the Activate At boxes, specify the time you want the tactical graphic to be visible.
   c. In the Deactivate At boxes, specify the time you want the tactical graphic to be hidden.
   d. Optionally, if you selected Use Date and Time of Day, you can select the Ignore Year check box to use only the month, date, and time of day. (The activation happens annually, at the same time each year.)
5. Click OK.

   If this is an areal graphic that will be active during specific times, it displays the activation and deactivation information in Zulu time.

48.3.  Allow Crossing

The Allow Crossing set data request allows human entities to walk across a crosswalk. When human characters plan a path that includes a crosswalk, when they arrive at the crosswalk, if crossing is allowed, they can cross the street. If it is not allowed, they stop and wait.

➤ **To set a crosswalk to be enabled or disabled:**

1. Select a crosswalk tactical graphic.
2. Choose **Set > Allow Crossing**. The Allow Crossing dialog box opens.
3. Select or clear the Allow Crossing check box.
4. Click OK.

48.4. Arrowhead Style

The Arrowhead Style set data request lets you set the type of arrowhead, location, and width for linear tactical graphics. For a description of arrowhead styles, see "47.10.6 Changing Linear and Areal Style Properties" on page 1033.

➤ **To set the arrowhead style:**

1. Select a linear tactical graphic.
2. Choose **Set > Appearance > Arrowhead Style**. The Arrowhead Style dialog box opens.

3. Select an option from the Arrow Style list.
4. Select an option from the Arrow Placement list.
5. Specify the width, in pixels, of the arrow.
6. Click OK.

48.5. Color

The Color set data request lets you change the color of linear and areal tactical graphics.

➤ **To set the color of a linear or areal tactical graphic:**

1. Select a tactical graphic.
2. Choose **Set > Appearance > Color**. The Color dialog box opens.
3. Select the color you want to use for this tactical graphic.
4. Click OK.

48.6. Fill Style

The Fill Style set data request lets you specify the fill for most areal tactical graphics. The fill can be horizontal, vertical, or diagonal lines, or various degrees of opacity.

➤ **To set the fill style of an areal tactical graphic:**

1. Select a tactical graphic.
2. Choose **Set > Appearance > Fill Style**. The Fill Style dialog box opens.
3. Select an option from the Fill Style list.
4. Click OK.

48.7. Force

You can change the force of a tactical graphic. For details, see "40.32. Force" on page 879.

48.8. Line Width

The Line Width set data request lets you set the width of linear and areal tactical graphics.

➤ **To set the color of a linear or areal tactical graphic:**

1. Select a tactical graphic.
2. Choose **Set > Appearance > Line Width**. The Line Width dialog box opens.
3. Type a value, in pixels, in the Line Width box.
4. Click OK.

48.9. Notify Level

You can set the notification level for console messages without opening the Information dialog box. For details, see "40.55. Notify Level" on page 890.

48.10. Pen Style

The Pen Style set data request lets you specify the type of line used to draw linear and areal tactical graphics. This includes solid lines and varies combinations of dashes and dots.

➤ **To set the pen style of a linear or areal tactical graphics:**

1. Select a tactical graphic.
2. Choose **Set > Appearance > Pen Style**. The Pen Style dialog box opens.
3. Select an option from the Pen Style list.
4. Click OK.



49. Remote Graphics

Remote graphics are graphics that are drawn onto the scene by remote programs that use the VR-Vantage Remote Draw API. This chapter explains how to display them in VR-Forces.

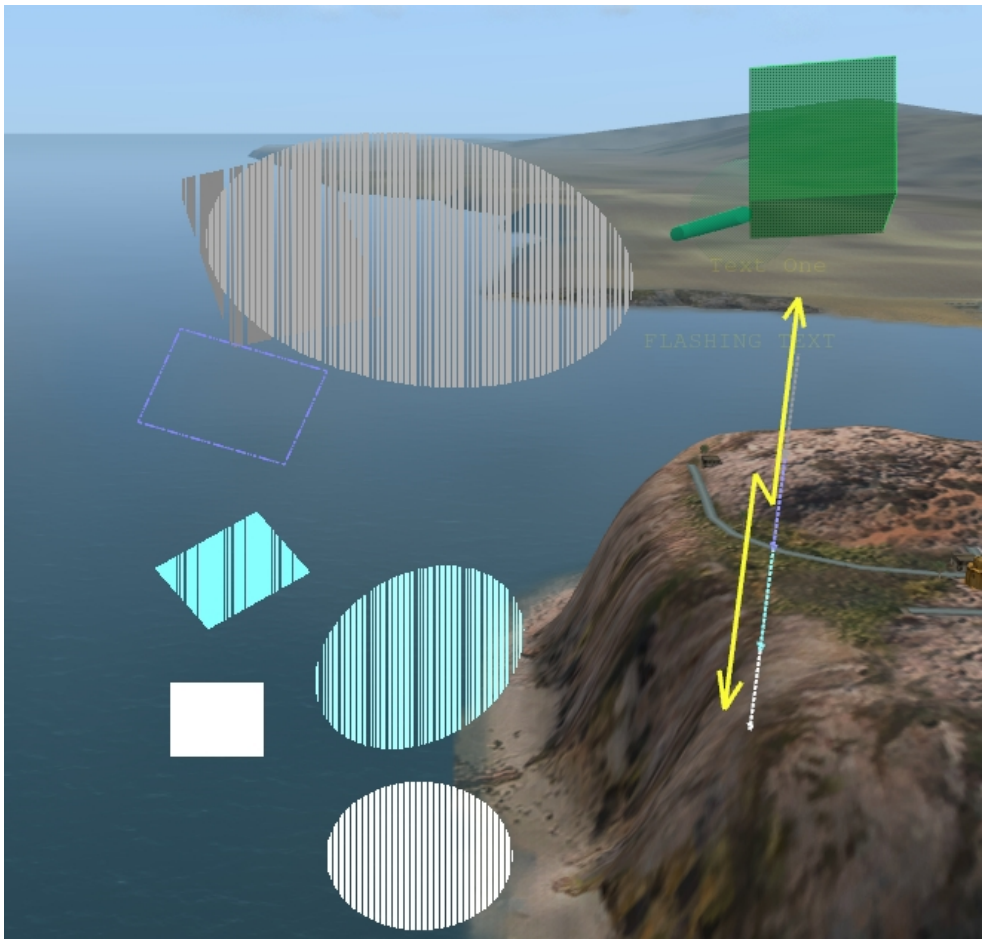
49.1. Displaying Remote Graphics 1044

49.1.1 Viewing a List of Remote Graphics 1045

49.1. Displaying Remote Graphics

VR-Forces can display graphics sent from other applications using the VR-Vantage Remote Draw API (Figure 368).

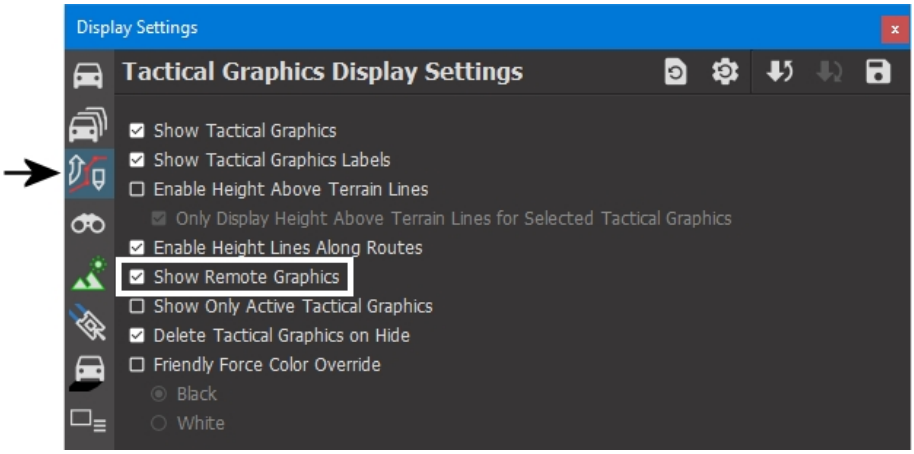
Figure 368. Remote graphics



➤ **To enable or disable the display of remote graphics:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Tactical Graphics Display Settings page (Figure 369).

Figure 369. Tactical Graphics Display Settings page



3. Select or clear the Show Remote Graphics check box.

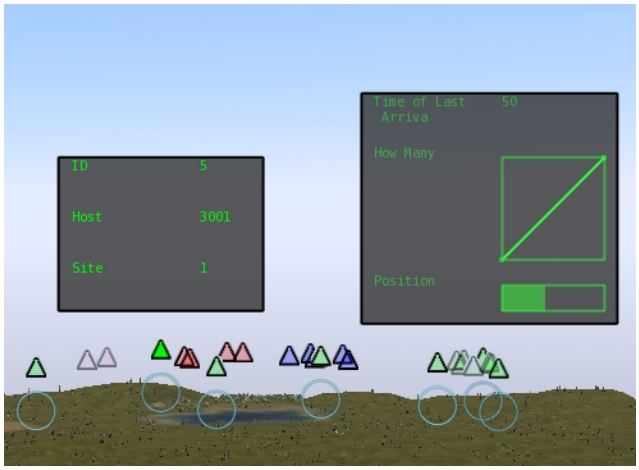
> You can also enable and disable remote graphics by choosing **Settings > Remote Graphics**.

49.1.1 Viewing a List of Remote Graphics

Remote graphics can be organized into sets. There is no fixed organization. Labeling of sets is entirely at the discretion of the programmer who develops the application that generates the graphics. The Remote Graphics Panel displays remote graphics in a hierarchy by set.

The VR-Vantage Remote Draw API lets developers send data about objects as attribute:value pairs, for example Speed:30 KM. The attributes are organized by type using attribute group templates. You can enable and disable the display of attribute information by group and by individual attribute. Figure 370 illustrates the attributes for the Entity and Packet attribute group templates displayed in Figure 371.

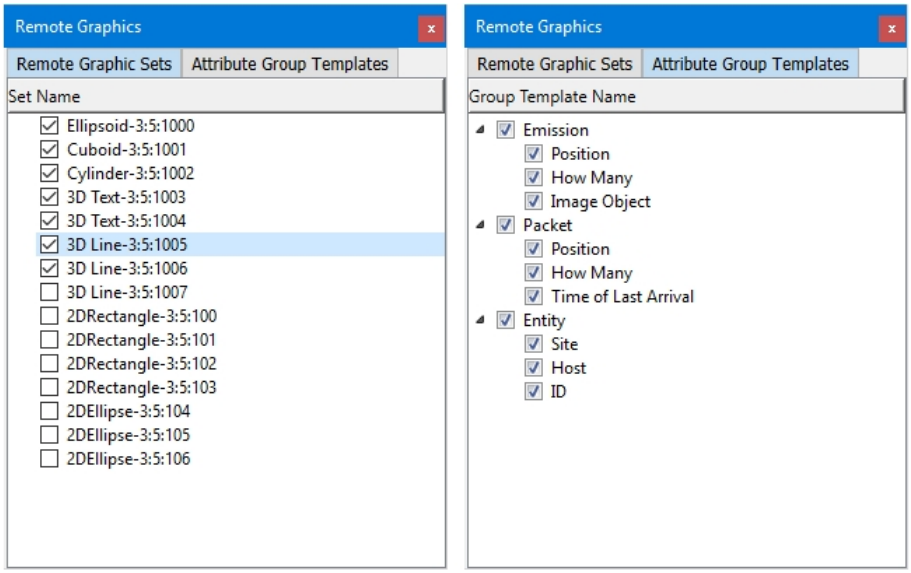
Figure 370. Remote graphics attribute group display



➤ **To view a list of remote graphics:**

1. Choose **View > Remote Graphics Panel**. The Remote Graphics Panel displays (Figure 371). The Remote Graphics Sets tab lists remote graphics by set.

Figure 371. Remote Graphics Panel



2. To hide or display a remote graphic set or individual graphic, check or clear its check box in the Remote Graphics Sets list.
3. To view Attribute Group Templates, select the Attribute Group Templates tab. To see the attribute display, mouse over an object for which display data is available.



X. Visual and Audio Effects

VR-Forces has many visual effects that add to your understanding of the interactions between simulation objects. These include fire and detonation lines, sensor contact lines, communications lines, and emitter volumes. You can test for line-of-sight with the intervisibility feature. You can also display many different lighting effects.



50. Displaying Tactical Information and Effects

This chapter describes how to manage special visual effects that can be applied to simulation objects.

> Many of the effects described in this chapter apply to individual entities in Stealth mode. By default, simulation objects in aggregate-level scenarios do not use 3D models. Therefore, effects such as trailing effects, smoke clouds, and fire and detonation effects are not supported.

50.1. Displaying Trailing and Decal Effects (3D Only) 1051
50.1.1 Specifying Models for Trailing Effects and Decal Effects 1052
50.1.2 Displaying Persistent Tracks 1053
50.2. Displaying Track Histories 1054
50.2.1 Configuring Track History Segments and Timeout 1055
50.2.2 Displaying HAT Lines for Track Histories (3D Only) 1056
50.2.3 Coloring an Entity's Track History 1056
50.2.4 Displaying a Terrain Profile for a Track History 1057
50.3. Viewing Fire and Detonation Effects 1057
50.4. Viewing Fire and Detonation Lines 1058
50.5. Viewing Unit Combat Graphics 1059
50.6. Visualizing Tasks 1060
50.6.1 Displaying Task Visualizations for an Entity 1062
50.6.2 Pinning Task Visualizations for an Entity 1062
50.7. Displaying Sensor Contact Lines 1063
50.7.1 Enabling the Display of Sensor Contact Lines 1063
50.7.2 Specifying Which Simulation Objects Display Sensor Contact Lines 1064
50.7.3 Configuring Sensor Contact Lines 1065
50.8. Displaying Threat and Sensor Range Rings 1065
50.8.1 Enabling and Disabling the Display of Range Rings 1067
50.8.2 Pinning Range Rings for a Simulation Object 1068
50.8.3 Configuring the Display of Range Rings 1068

50.8.4 Configuring Display of Weapon System Ranges 1070
50.9. Displaying Laser Designators 1070
50.10. Displaying Supply Lines 1071
50.11. Displaying Tactical Smoke 1072
50.12. Displaying Electromagnetic Emissions 1072
50.13. Displaying Radio Communication Lines 1073
50.13.1 Enabling Radio Communications Graphics 1074
50.13.2 Configuring the Lifetime and Display Mode 1076
50.13.3 Configuring the Color of Radio Communication Lines 1076
50.14. Enabling and Disabling Wheel Rotation 1078

50.1. Displaying Trailing and Decal Effects (3D Only)

VR-Forces supports trailing effects such as contrails for missiles and aircraft and dust clouds for vehicles. It also supports decal effects, such as tire tracks, footprints, and wakes. In essence, trailing effects are effects that are dispersed through the air, while decal effects are effects that are printed on the ground.

Trailing effects and decal effects are properties of an observer mode. You can enable and disable them on a per-observer basis.

Figure 372 illustrates some trailing effects.

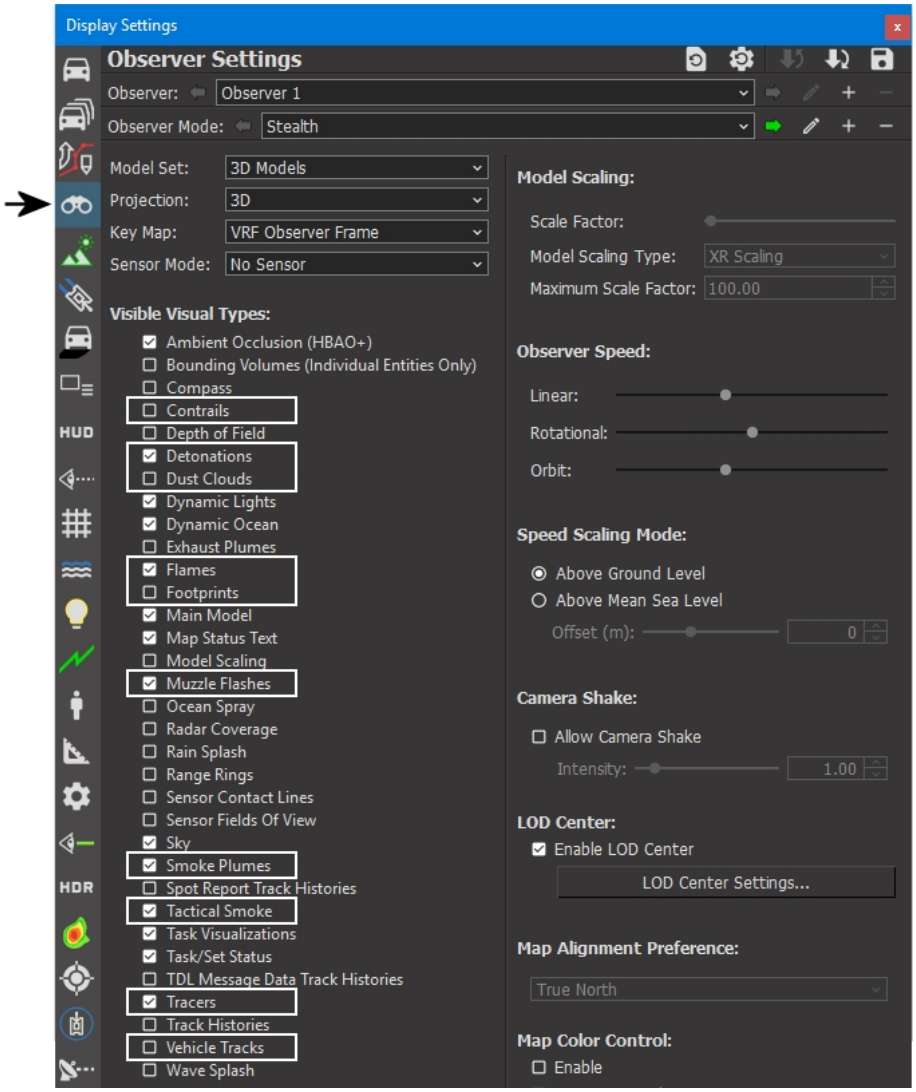
Figure 372. Vehicle tracks, dust cloud



➤ **To enable or disable trailing effects or decal effects:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373).

Figure 373. Observer Settings page



3. Select or clear the check box for each effect that you want to change.

> You can also enable and disable trailing and decal effects by choosing **Observer > *effect***.

50.1.1 Specifying Models for Trailing Effects and Decal Effects

Trailing effects and decal effects are configured through model definitions. (For the purposes of configuration, trailing effects are categorized as decals.) Each effect must have a model definition that uses a decal schema. Then, in an entity's entity definition, you configure a visualizer for the effect. For details about creating model definitions, please see "Creating and Editing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.

50.1.2 Displaying Persistent Tracks

When displaying tracks, the default for decal effects of vehicle's tracks is to fade after about 20 seconds. You can specify that those tracks should persist on the terrain rather than fading out. This is primarily intended for use with aerial surveillance scenarios. Tracks must be enabled for the persistent tracks to display. See "50.1. Displaying Trailing and Decal Effects (3D Only)" on page 1051.

For persistent tracks to display on the terrain, you must draw a Persistent Tracks Area and then create tracks in that area. There are two ways to create persistent tracks: set vehicles to move through the area, or draw tracks in that area.

➤ **To permanently display vehicles' tracks in an area:**

1. Open the Create Object Palette.
2. Click the Terrain filter ( ).
3. Select Persistent Tracks Area.
4. Click to place the vertices of the persistent tracks area. Double-click when you place the final vertex. The area is created.

   While the scenario is running, any ground vehicles that move through the area will leave permanent, persistent decal effects.

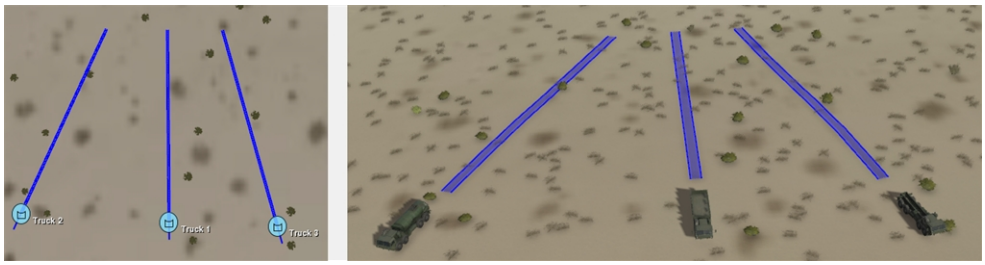
➤ **To draw persistent tracks:**

1. Choose **Simulation > Dynamic Terrain > Add Persistent Tracks**. The Add Persistent Tracks dialog box opens.
2. Click the Draw Track button. The Create Route dialog box opens.
3. On the terrain, draw a route where you want the tracks to display. Double-click to place the final point or click Create. You return to the Add Persistent Tracks dialog box.
4. From the Track Type list, choose the type of tracks you want to create:
   - Inline Wheeled. Motorcycles, bicycles, and other vehicles where one wheel is behind the other.
   - Wheeled. Vehicles with two wheels per axle.
   - Dual Wheeled. Vehicles with four wheels on one or more axles.
   - Tracked. Tanks and other vehicles that move using continuous track, caterpillar track, or treads.
5. From the Track Depth list, choose Light, Medium, or Heavy. Light tracks are less visible than Medium tracks, and likewise Medium tracks are less visible than Heavy tracks.
6. Click OK.

50.2. Displaying Track Histories

Track histories allow you to track the paths of simulation objects during an exercise, including the path of munitions as they travel from the shooter to the target. When track histories are enabled, each simulation object leaves a track ribbon. The ribbon's default color is based on the simulation object's force. Figure 374 illustrates track histories.

Figure 374. Track history - 2D and 3D



Track histories are a property of an observer mode. You can enable and disable them on a per-observer basis.

- Track histories have a significant performance and memory cost. For simulations with large simulation object counts, turning track histories off can improve performance and memory usage.
- If a ground entity is traveling over uneven terrain, the track history may appear broken. If you want to see it as a complete, solid line, you must edit the entity's definition and change the Track History's **allowocclusion** setting to false. You must apply this setting before connecting to the simulation. See the chapter on *Element Definitions* in the *Adding Content to MAK ONE Applications Reference Manual*.
- If you reposition a simulation object, any existing track history is removed and a new one starts to draw.

➤ **To enable or disable display of all track histories:**

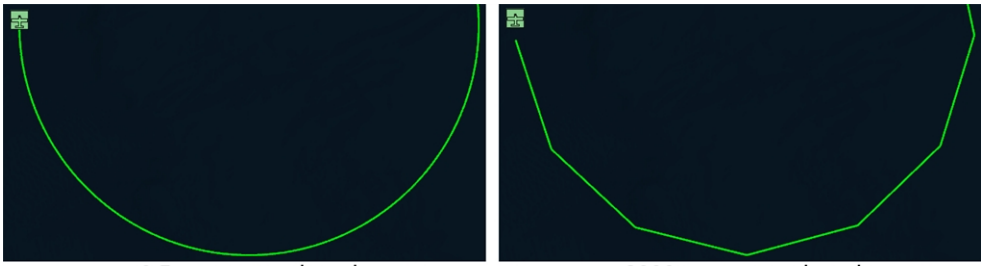
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373 on page 1052).
3. Select the observer mode for which you want to change the Track Histories setting.
4. Select or clear the Track Histories check box.

You can also enable or disable track histories by choosing **Observer > Track Histories** or clicking the Track Histories button ( ) on the Observer Settings Toolbar.

50.2.1 Configuring Track History Segments and Timeout

You can configure how long track history ribbons are drawn. A track history is composed of a series of segments joined together to form a line. You can configure how long each segment is and how many segments at a time can be displayed. Drawing shorter segments improves the accuracy of the track history, but can degrade performance. Increasing the maximum number of segments provides a longer history, but can clutter up the screen if there are many simulation objects and can also affect performance. Figure 375 illustrates the visual result of drastically changing the segment length. The left track history uses the default segment length of 0.5 meters. The right track history has a segment length of 3000 meters.

Figure 375. Track history segment length



When a track history reaches the maximum number of segments configured, it starts dropping segments from the beginning of the history (the oldest segments).

You can specify that track histories have an infinite length. In this case they never drop segments.

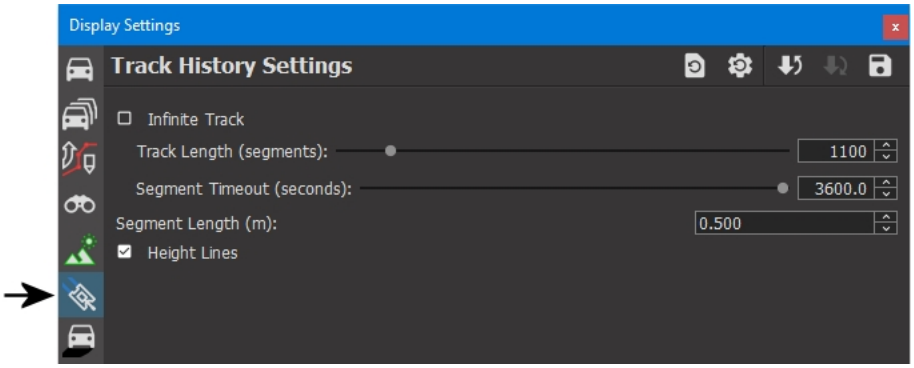
> ⚠ If you specify infinite length for track histories, it is possible that you could eventually use up all the memory in your computer and the program would crash.

You can also specify a timeout value for track histories. The total length of a track history is determined by a combination of the oldest segment creation time, the segment length, and the number of segments.

➤ **To configure track histories:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Track History Settings page (Figure 376).

Figure 376. Track History Settings page



3. If you want track histories to have an indefinite length, select the Infinite Track check box.
4. To change the number of segments that can be displayed for a track history, adjust the Track Length slider, or enter a value in the box.
5. To change the segment timeout, adjust the Segment Timeout slider. The default is 3600 seconds (60 minutes).
6. To specify the length of each track history segment, enter a value in the Segment Length box.

50.2.2 Displaying HAT Lines for Track Histories (3D Only)

You can choose whether to display height-above-terrain (HAT) lines for track histories in the scenario. These thin, light-colored lines display in regular intervals, pointing towards the terrain to help you visualize the height. There is no terrain intersection; if the route is above water, the HAT lines can go below sea level.

> To configure the appearance of the HAT lines, edit the Track History visual definition for the entity's element definition and specify the values for **enableheightlines**, **heightlinescolor**, and **heightlinesspacing**. See the *Element Definitions* chapter in the *Adding Content to MAK ONE Applications Reference Manual* to learn more.

➤ **To enable or disable HAT lines for track histories:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Track History Settings page (Figure 376 above).
3. Select or clear the Height Lines check box.

50.2.3 Coloring an Entity's Track History

You can change the color of an entity's track history. (By default, the track uses the color assigned to the entity's force.) The color you assign is visible only to you in your local GUI and does not persist in later sessions.

➤ **To change an entity's track history color:**

1. Right-click the entity and select **Color Track History**.
2. In the submenu, select a color or **Custom Color**.
3. If you selected **Custom Color**, choose a swatch from the basic colors, click a spot on the palette, specify a shade using its HTML color code, or click **Pick Screen Color** to use a color you see on the screen.
4. To return the entity to its default color, select **Color Track History > Clear Override**.

50.2.4 Displaying a Terrain Profile for a Track History

You can display a terrain profile for a track history. (For complete information about terrain profiles, see "Chapter 65. Using Terrain Profiles" on page 1265.)

> The track history terrain profile does not display the track history that is drawn in the window when you enable track histories. It creates an independent track that starts when you open the terrain profile window.

➤ **To display a terrain profile for a track history:**

1. Select the simulation object whose track history terrain profile you want to see.
2. Choose **Objects > Terrain Profile > Track History Profile**. The Terrain Profile dialog box opens. It updates as the simulation object moves.

50.3.  Viewing Fire and Detonation Effects

VR-Forces can display graphic effects when munitions are fired and detonated. They are not supported in 2D observer modes.

- Flaming effects.
- Smoke plumes.
- Detonations.
- Muzzle flashes.
- Tracers. Tracers are displayed if the munition in a fire interaction is a tracer munition and the munition is mapped to a tracer model, and the munition must be direct fire. You can configure munitions in the Simulation Object Editor. For details, see "70.5. Creating and Editing Munitions" on page 1393. (Entities with weapons that support tracer munitions must enable tracer use for tracers to display. For details about enabling tracer use, see "40.92. Tracer Use" on page 907.)

➤ **To enable fire and detonation effects:**

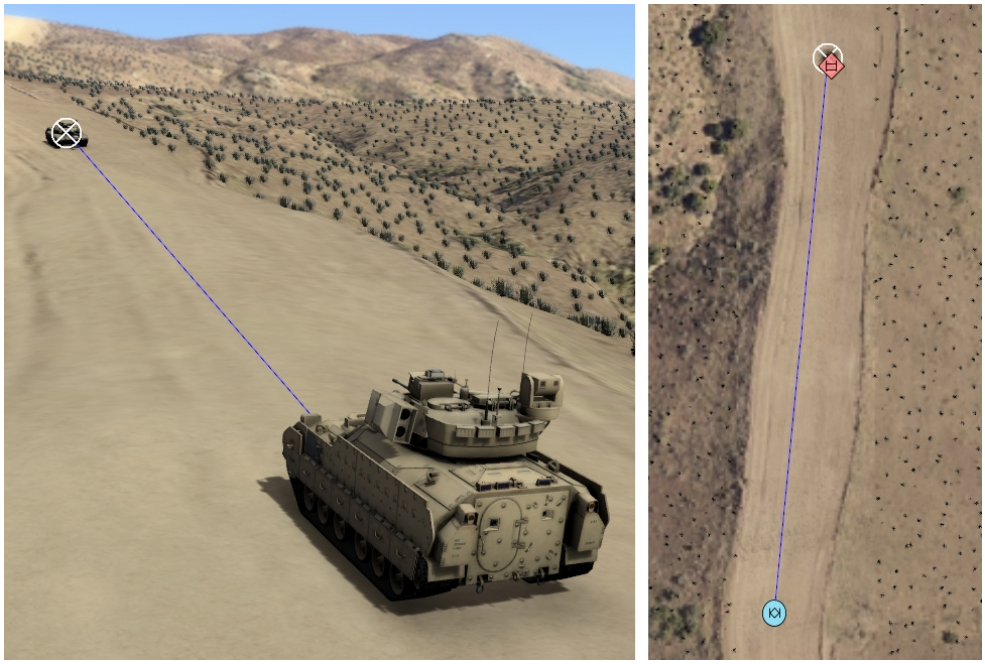
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373 on page 1052).
3. Select or clear the appropriate check boxes.

> You can also enable or disable display of a fire or detonation effect by choosing **Observer >** *effect*, where *effect* is Flames, Smoke Plumes, Detonations, Muzzle Flashes, or Tracers.

50.4. Viewing Fire and Detonation Lines

If fire and detonation lines are enabled, when an entity fires a munition, a line is drawn between the entity and the location of the detonation. The line is colored for the force of the shooter (Figure 377).

Figure 377. Fire line and detonation - 3D and 2D



A detonation is identified by a circle with an X through it. The detonation color indicates the result of the detonation:

- Entity impact – white.
- Ground impact – gray.
- Airburst – black.
- Other – white.

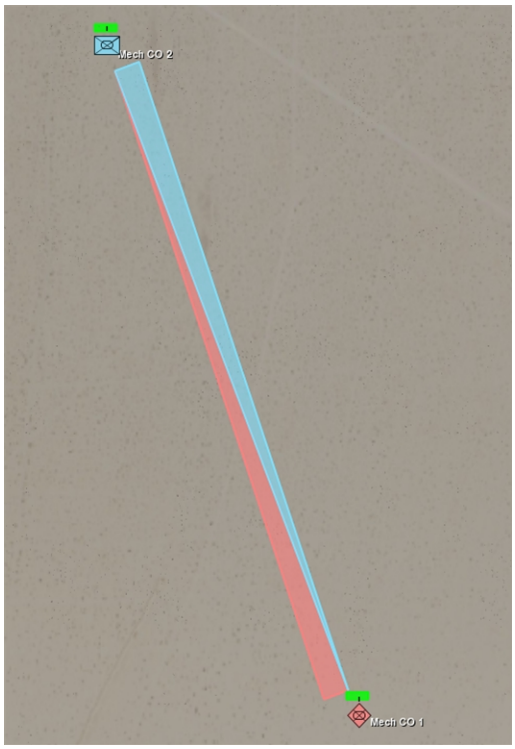
➤ To enable or disable display of fire and detonation lines, choose **Settings > Fire / Detonate Lines** or click the Fire/Detonate Lines button ( ) on the Display Settings Toolbar.

50.5. Viewing Unit Combat Graphics

When simulation objects in aggregate-level scenarios engage in combat, they display two types of graphics, attack indicators, and attack animations. Attack indicators are triangular areas that show the direction a unit is attacking. They are colored with the force of the attacker. Attack animations show where a simulation object is being attacked. The footprint sector where the attack is taking place blinks from green to white.

The graphics persist until the attack is over.

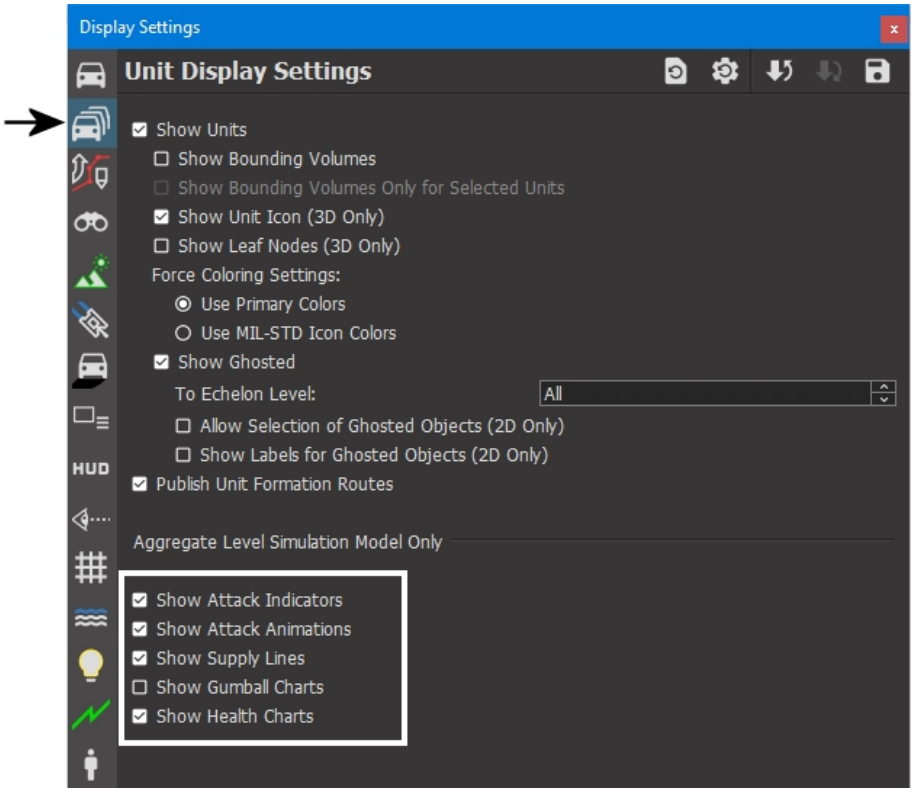
Figure 378. Attack indicators



➤ **To enable or disable unit combat graphics:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 379).

Figure 379. Unit Display Settings page



3. To enable or disable attack indicators, select or clear the Show Attack Indicators check box.
4. To enable or disable attack animations, select or clear the Show Attack Animations check box.

50.6. Visualizing Tasks

VR-Forces can display graphics that help to show the task a simulation object is executing. Figure 380 and Figure 381 illustrate task visualization lines in Plan View mode and Stealth mode that show an entity moving to a location, and also an aircraft entity moving to a waypoint with its estimated time of arrival according to the simulation clock.

Task visualizations are not available for some tasks.

Task visualizations are supported only for individual entities, not for units. Additionally, the ETA feature is supported for all entities except ground vehicles in entity-level scenarios.

To maintain adequate GUI performance, task visualizations have limits:

- No more than 10 simulation objects can show task visualizations at the same time.
- The maximum number of segments in a task visualization line is 80. If a simulation object is following a route or series of points that requires more than 80 segments to visualize, VR-Forces samples the points in the line. The resulting visualization line will not be as precise as the actual path the object is following, but still provides a general sense of its intent.

If either of these conditions is met, VR-Forces prints messages when the notification level is set to verbose.

You can specify that tasks be visualized for the selected simulation object. You can also specify that tasks be visualized on a per simulation object basis. The per-simulation object setting is saved as part of the scenario.

Figure 380. Task visualization of human entity moving to location



Figure 381. Task visualization of a fixed wing entity moving to a waypoint



➤ To enable or disable task visualization lines, choose **Observer > Task Visualizations** or click the Task Visualizations button () on the Observer Toolbar.

50.6.1 Displaying Task Visualizations for an Entity

If task visualization is enabled, by default they are displayed for the selected entity. You can disable automatic display of task visualizations for the selected entity. If this setting is disabled, you can still show a task visualization by pinning it to an entity, as described in "50.6.2 Pinning Task Visualizations for an Entity" below.

➤ **To specify that the selected simulation object show task visualization lines:**

1. Enable task visualizations.
2. Select an entity.
3. Choose **Settings > Display**. The Display Settings dialog box opens.
4. Select the Entity Display Settings page.
5. Select the Show Task Visualizations for Selected Entity check box. Clear the check box to disable them.

50.6.2 Pinning Task Visualizations for an Entity

By default, when task visualizations are enabled, they are only displayed for the selected entity. You can pin task visualization to an entity so that they are displayed even if it is not selected. This is the only way to have multiple entities visualize their tasks at the same time. You can pin a task visualization and display it even if the Show Task Visualization for Selected Entity check box on the Entity Display Settings page is cleared.

➤ **To pin task visualizations for a simulation object:**

1. Select the simulation object for which you want to visualize tasks.
2. Choose **Objects > Decorations > Pin Task Visualizations**.

50.7. Displaying Sensor Contact Lines

You can display lines between a simulation object and the objects it detects with its sensors (Figure 382 below). Display of sensor contact lines requires two levels of control. Display of sensor contact lines is an observer-specific setting. They must be enabled in the observer mode to be displayed.

When display of sensor contact lines is enabled, you can specify that sensor contact lines be shown for specific simulation objects, for the selected simulation object, or for both. If you enable display of sensor contact lines for a specific simulation object, the setting is saved with the scenario.

You can also configure the width and color of sensor contact lines.

Figure 382. Sensor contact lines



50.7.1 Enabling the Display of Sensor Contact Lines

Regardless of other settings, you must enable the display of sensor contact lines in the observer to see them.

➤ **To enable the display of sensor contact lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373 on page 1052).
3. In the settings list, select Sensor Contact Lines.

You can also enable or disable sensor contact lines by choosing **Observer > Sensor Contact Lines** or clicking the Sensor Contact Lines button ( ) on the Observer Settings Toolbar.

50.7.2 Specifying Which Simulation Objects Display Sensor Contact Lines

Assuming that the display of sensor contact lines is enabled, you can specify that particular simulation objects always display them and you can specify that simulation objects display sensor contact lines if they are selected.

➤ **To display sensor contact lines for a simulation object whether it is selected or not:**

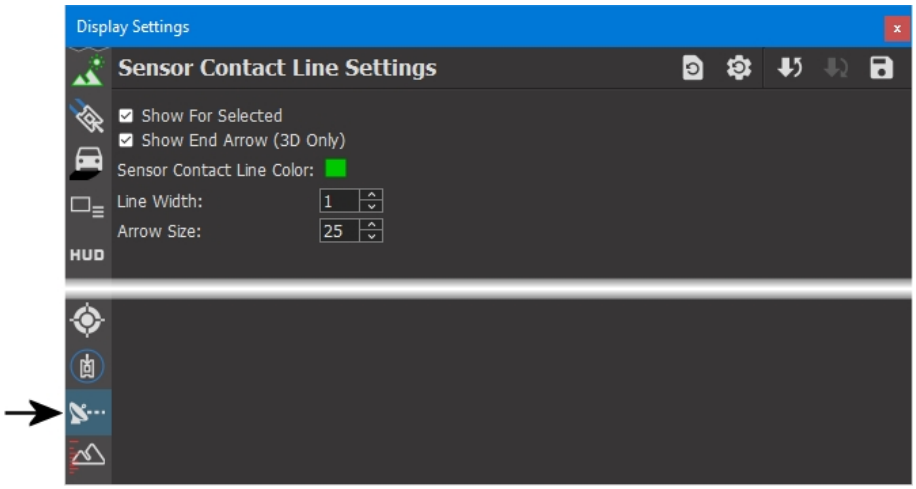
1. Select the simulation objects for which you want to enable sensor contact lines.
2. Choose **Objects > Decorations > Pin Sensor Contacts**.

ⓘ This setting is saved with the scenario.

➤ **To enable display of sensor contact lines by the selected simulation objects:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Sensor Contact Line Settings page (Figure 383).

Figure 383. Sensor Contact Settings page



3. Select or clear Show For Selected.

ⓘ This setting is saved with the GUI settings.

Removing Sensor Contact Line Pins

➤ **To remove the sensor contact line pin:**

1. Select a simulation object that has sensor contact lines pinned.
2. Choose **Objects > Decorations > Pin Sensor Contacts**. The track history pin is removed.

➤ To remove track history pins for all entities, choose **Objects > Decorations > Remove All Pin Sensor Contacts**.

50.7.3 Configuring Sensor Contact Lines

You can set the color, line width, and arrow size of sensor contact lines.

➤ **To configure sensor contact lines:**

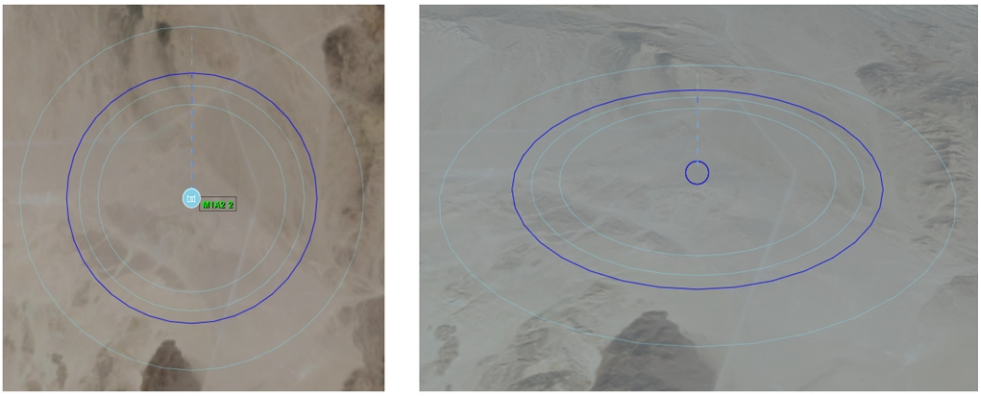
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Sensor Contact Settings page (Figure 383 on the previous page).
3. Change the values as desired.

50.8. Displaying Threat and Sensor Range Rings

You can display a simulation object's threat range – the area within which its armaments are effective. The threat range is displayed as a circle, with a line indicating the entity's heading (Figure 384). A ring displays for each weapon system that the entity has. For example, in Figure 384, the M1A2 entity has an inner ring for its 62 mm machine gun and an outer ring for its 120 mm main gun.

You can also display range rings for a simulation object's sensors.

Figure 384. Range rings (entity-level scenario)

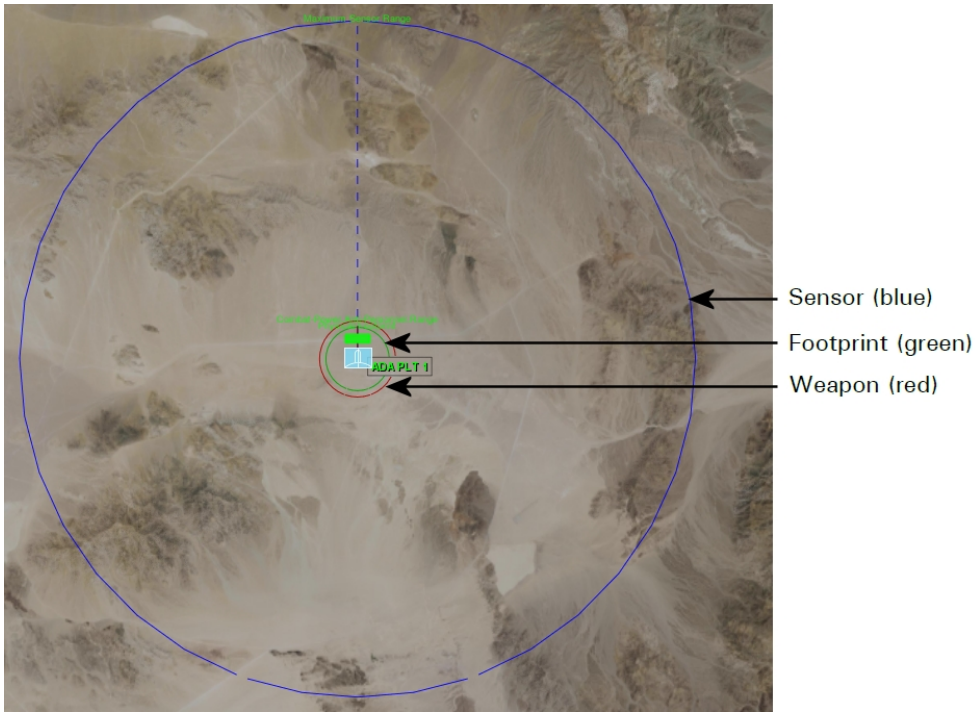


2D 3D

Simulation objects in aggregate-level scenarios display range rings that show their combat effectiveness and a footprint that represents the physical area that the unit occupies (Figure 385). Unit range rings have breaks to show the front, rear, left flank, and right flank sectors.

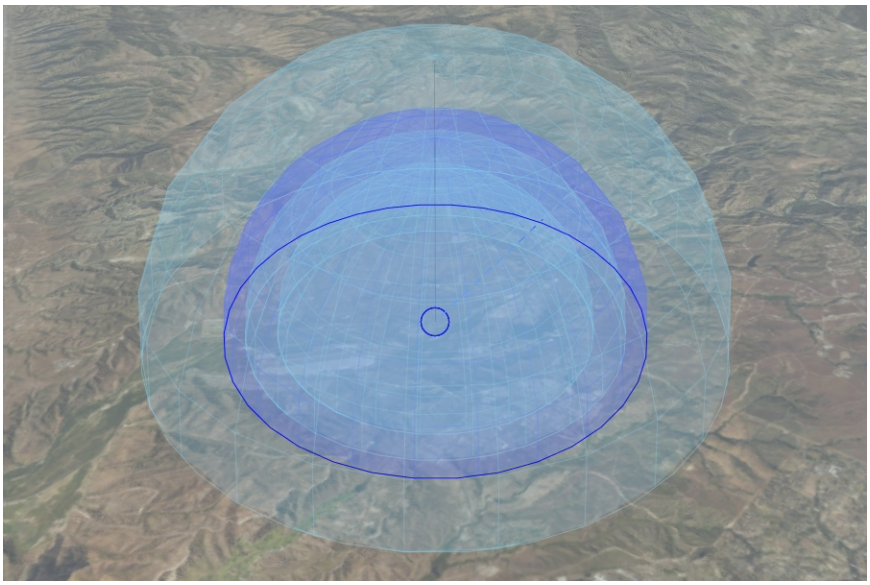
On small databases, a simulation object's range ring may extend beyond the edge of the terrain display.

Figure 385. Range rings (aggregate-level scenario)



You enable range rings globally. They are an observer-specific option. You display range rings on a per-entity basis. You can globally configure the color of rings, their transparency, and the display of labels. In 3D views, you can specify whether to display a range sphere (Figure 386).

Figure 386. Range ring sphere

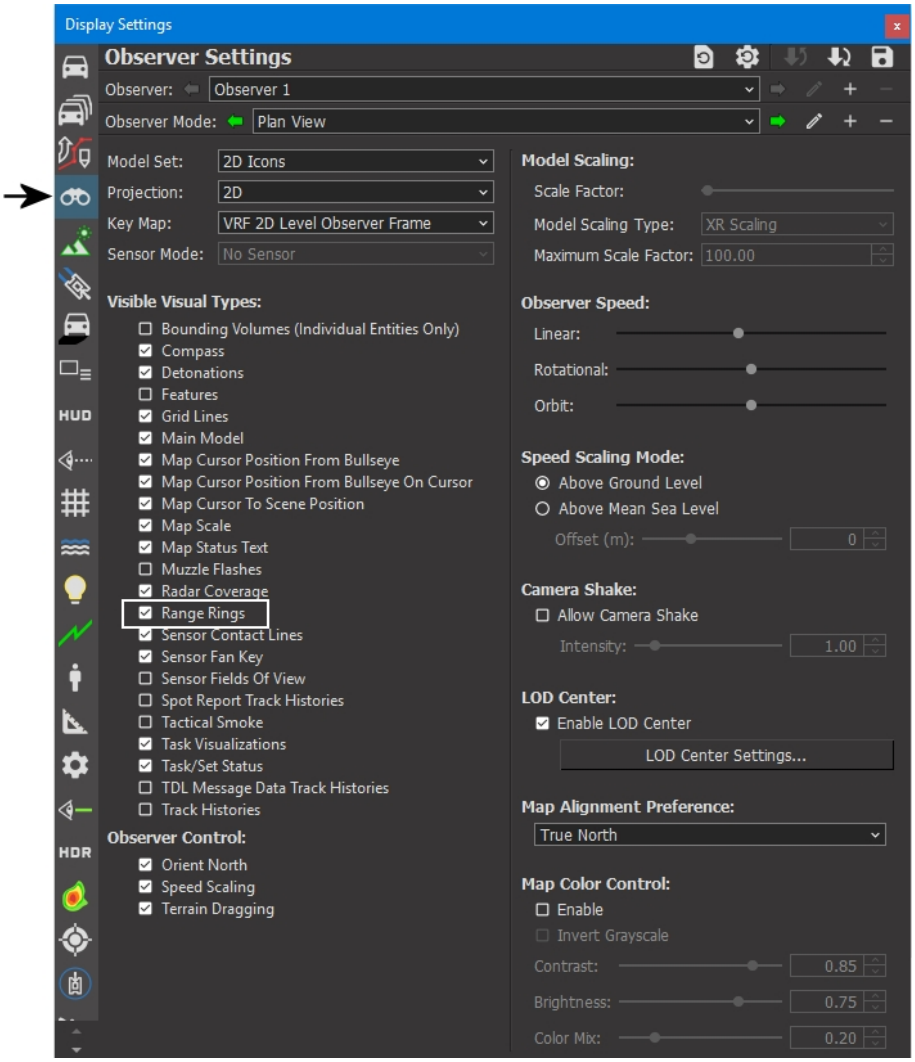


50.8.1 Enabling and Disabling the Display of Range Rings

➤ **To enable or disable the display of range rings:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 387).

Figure 387. Observer Settings page



3. In the list of options, select or clear the Range Rings check box.

> You can also enable and disable range rings by choosing **Observer > Range Rings** or by clicking the Range Rings button ( ) on the Observer Settings Toolbar.

50.8.2 Pinning Range Rings for a Simulation Object

You display range rings on a per-entity basis. By default, if range rings are enabled, when you select an object, its range rings are displayed as long as the object is selected. You can pin range rings to an object so that they are displayed even if the object is not selected.

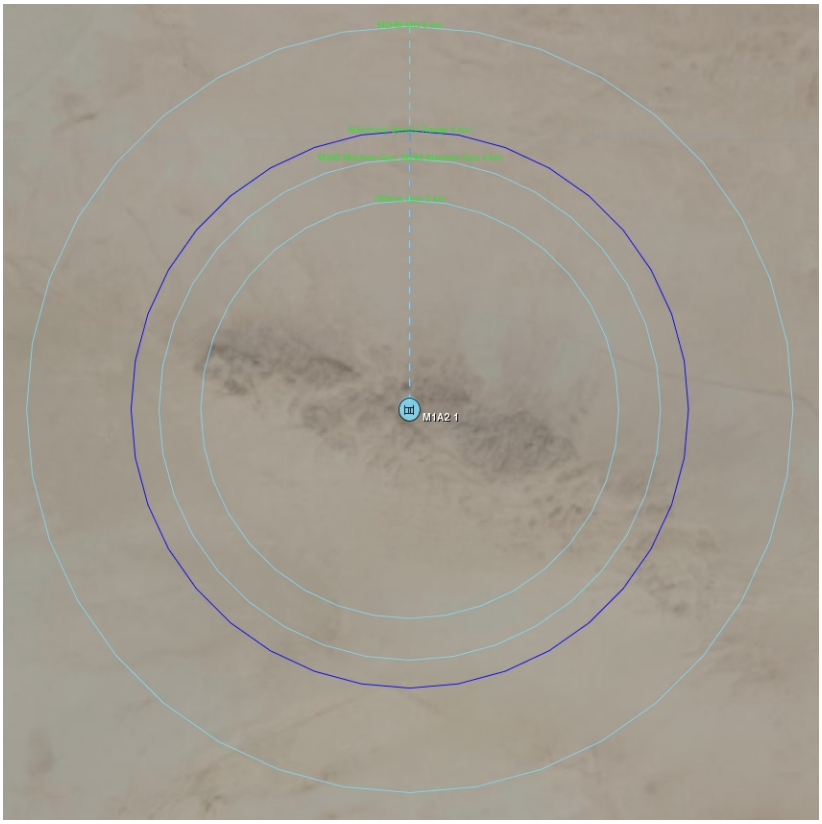
➤ **To pin range rings for a simulation object:**

1. Enable range rings, as described in "50.8.1 Enabling and Disabling the Display of Range Rings" on the previous page.
2. Select a simulation object.
3. Choose **Objects > Decorations > Pin Range Rings**.

50.8.3 Configuring the Display of Range Rings

You can configure the color of range rings and their transparency by weapon type and sensor type. You can also configure the display of range ring labels (Figure 388). A label can display the entity name, weapon type, and range of the weapon.

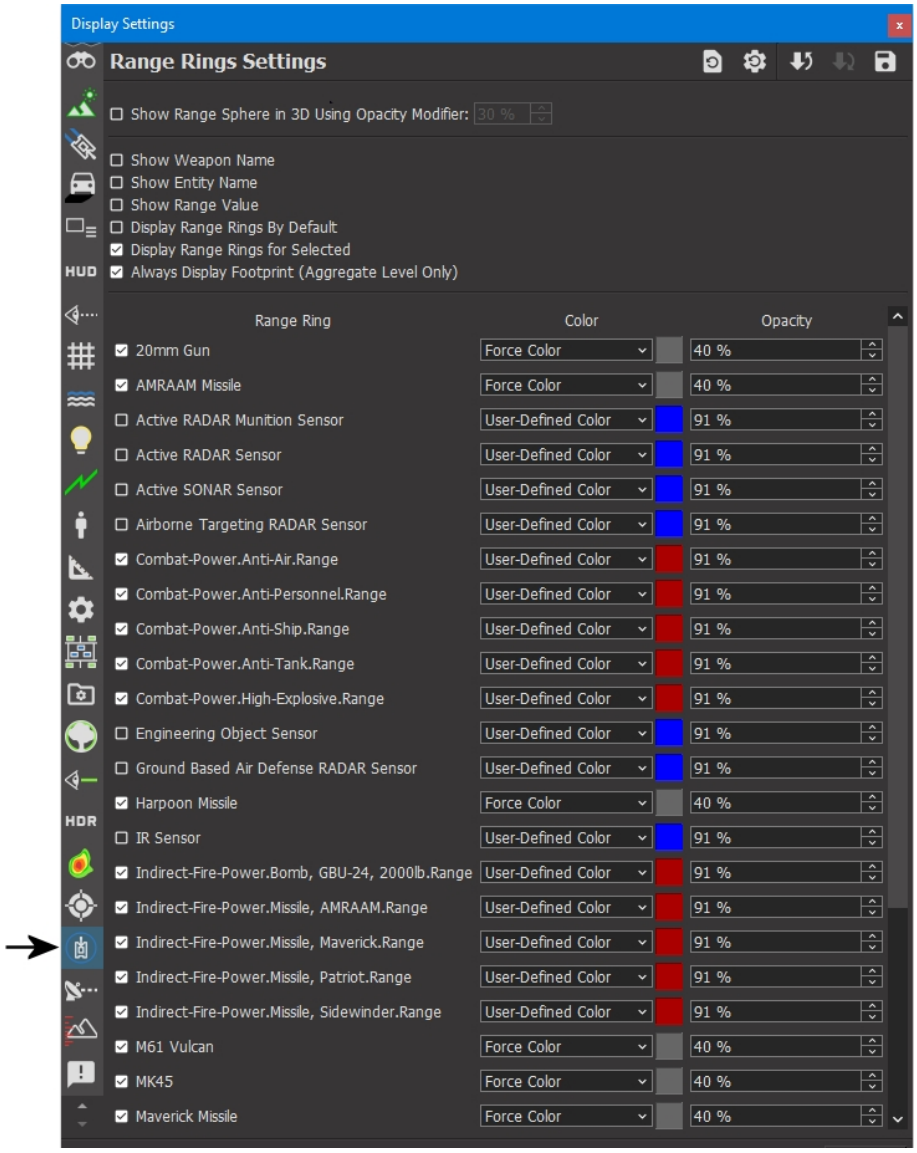
Figure 388. Range ring labels



➤ **To configure the display of range rings:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Range Rings Settings tab (Figure 389).

Figure 389. Range Rings Settings page (*AggregateLevelBase.sms*)



3. To display a sphere in 3D views, select the Show Range Sphere in 3D Using Opacity Modifier check box. Optionally, change the opacity modifier value.
4. To display the type of sensor or weapon system, select the Show Weapon Name check box.
5. To display the name of the simulation object, select the Show Entity Name check box.
6. To show the range of the sensor or weapon system, select the Show Range Value check box.
7. To automatically display range rings when a simulation object is created or discovered on the network, select Display Range Rings by Default.

> If you select this option, when you load a scenario, range rings are displayed for all simulation objects. When you select the option after loading a scenario, they display only for simulation objects that are created or discovered on the network after you selected the option.

8. To display range rings for a selected simulation object that has not enabled range rings, select Display Range Rings for Selected.
9. To always show the footprint of a simulation object, select Always Display Footprint.
10. To enable or disable display of range rings for a particular weapon system, select or clear the check box for that system.
11. To change the color used to display a range ring:
    a. For the type of weapon system or sensor whose color you want to change, select User-Defined Color in the Color column list. The color swatch next to the list changes.
    b. Click the color swatch. A color picker dialog box opens.
    c. Select the color you want to use.
    d. Click OK.
12. To change the opacity for a range ring, for the weapon system whose range ring opacity you want to change, change the value in the Opacity column.

50.8.4 Configuring Display of Weapon System Ranges

You can use range rings to display the range for weapon systems and publish them to the network as enumerations. If you create a new weapon system, you must add it to the list.

Range ring enumerations are stored in: *./appData/settings/enumerations/VRF-enums.xml*.

The range ring name and enumeration are defined in the weapon's sysdef file. For example:

```
(DtRwString
  (default-range-name "120mm Gun")
)
(DwRwInt
  (range-id 7)
)
```

50.9.  Displaying Laser Designators

You can display laser designation information if it is available. VR-Forces displays a green circle and the laser code (on top of the entity icon or model) (Figure 390).

Figure 390. Laser designators - 3D and 2D



➤ To display laser designators, choose **Settings > Laser Designators**.

ⓘ In the Visual Model Editors, you can configure the 2D and 3D Laser Designators visualizer definitions for entities to display a line, and you configure the appearance of the line. In the Entity Definition Editor, specify the desired values for the islaserlinevisualized, lasercodelodoutaltitude, laserlinestipplepattern, or laserlinecolor visual definition attributes.

50.10. Displaying Supply Lines

Simulation objects in aggregate-level scenarios can provide supplies to and receive supplies from other simulation objects. When a simulation object is supplying another entity you can display supply lines that show resupply is taking place.

➤ **To display supply lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Unit Display Settings page (Figure 379 on page 1060).
3. Select or clear Show Supply Lines.

50.11. Displaying Tactical Smoke

VR-Forces can display tactical smoke munitions fired by VR-Forces entities. You may want to disable tactical smoke if it obscures the observer's view. Also, if many entities are generating smoke, it could affect performance.

- Do not confuse tactical smoke and smoke plumes. Smoke plumes are a visual effect displayed when an entity is on fire. Tactical smoke visualizes a smoke munition that affects line-of-sight.
- If you disable display of tactical smoke, VR-Forces continues to simulate its existence in the sim engine and it continues to affect line-of-sight and IR sensors.

➤ **To enable tactical smoke:**

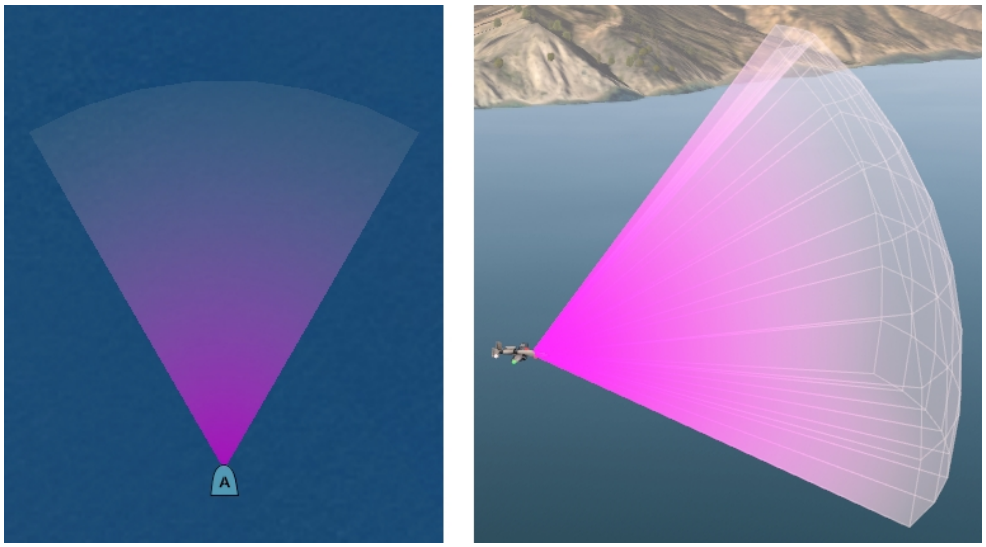
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373 on page 1052).
3. Select or clear the Tactical Smoke check box.

You can also enable or disable display of a fire or detonation effect by choosing **Observer > Tactical Smoke**.

50.12. Displaying Electromagnetic Emissions

VR-Forces can display electromagnetic emissions (emitter volumes) for entities that have emitter systems (Figure 391). You can configure the colors used to represent emitter frequencies and you can configure the size of the emitter volumes. (For details about configuring emitter volumes, see "56.1. Emitter Volumes Overview" on page 1136.)

Figure 391. Emitter volumes - 2D and 3D

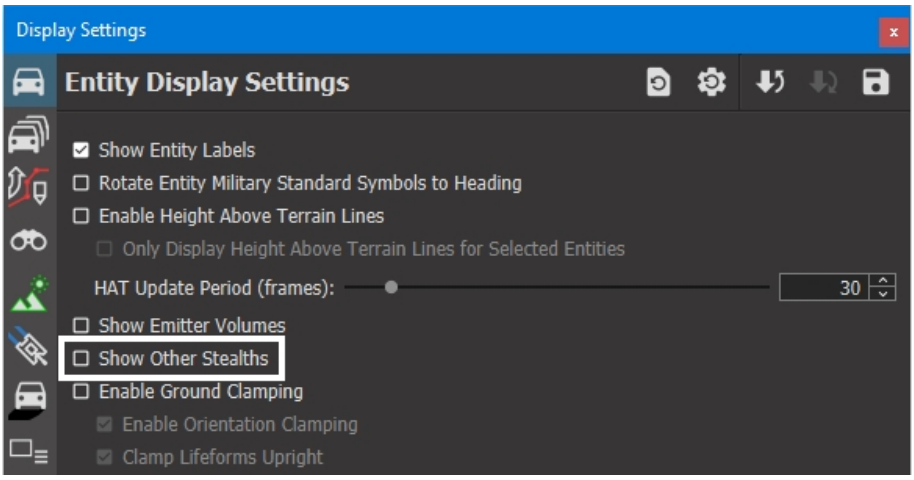


Display of emitter volumes only affects entities that have emitter systems.

➤ **To enable or disable display of electromagnetic emissions:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 392).

Figure 392. Entity Display Settings page



3. Select or clear the Show Emitter Volumes check box.

You can also enable or disable emitter volumes by choosing **Settings > Emitter Volumes**, or clicking the Emitter Volumes button ( ) on the Display Settings Toolbar.

50.13. Displaying Radio Communication Lines

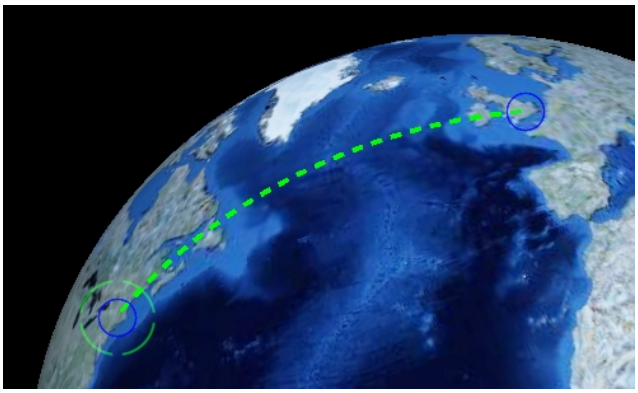
VR-Forces can display squawk indicators when a simulation object sends a signal interaction. It can also display communication lines between simulation objects when a VR-Forces simulation object (or a simulation object in an application built with the appropriate MAK libraries) sends a radio message to another simulation object. You can configure a communication line's color and lifetime. Figure 393 displays squawk indicators and a line-of-sight communication line.

Figure 393. Radio communication line and squawk indicators



For communications over large distances, a line-of-sight communication line would travel through the earth. Therefore, you can also display communication lines drawn with the curvature of the earth (Figure 394).

Figure 394. Curvature of the earth communication line



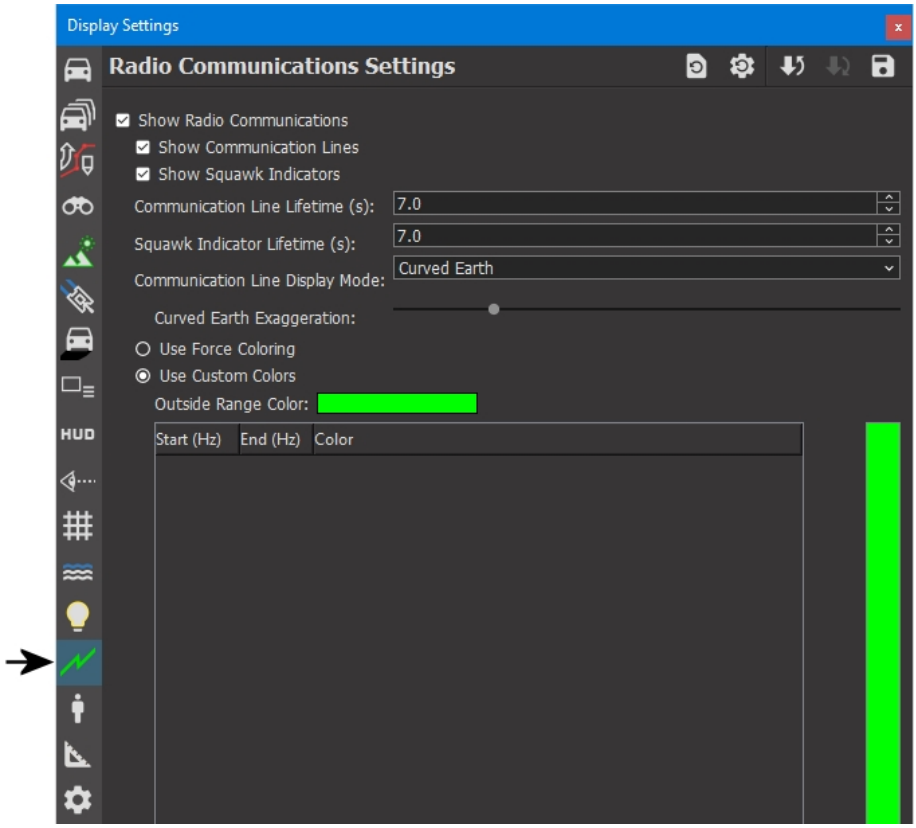
50.13.1 Enabling Radio Communications Graphics

You can globally enable or disable the ability to display radio communication lines. When they are enabled, you can enable and disable communication lines and squawk indicators separately.

➤ **To enable or disable radio communications graphics:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 395).

Figure 395. Radio Communications Settings page



3. Select or clear the Show Radio Communications check box.

> You can also enable and disable communication lines by choosing **Settings > Radio Communication Display** or by clicking the Radio Communications button ( ) on the Display Settings Toolbar.

➤ **To enable or disable communication lines:**

1. Enable radio communication lines.
2. Select or clear the Show Communication Lines check box.

➤ **To enable or disable squawk indicators:**

1. Enable radio communication lines.
2. Select or clear the Show Squawk Indicators check box.

50.13.2 Configuring the Lifetime and Display Mode

You can configure how long a radio communication line displays. You can also configure whether it displays using line-of-sight or curvature of the earth.

➤ **To configure the lifetime and display mode for radio communication lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 395 on the previous page).
3. To specify how long a communication line or squawk indicator should be displayed, enter a value, in seconds, in the Communication Line Lifetime and Squawk Indicator Lifetime fields.
4. To specify the display mode, select an option in the Communication Line Display Mode list.
5. If you select Curvature of the Earth, adjust the Curved Earth Exaggeration slider to change the degree to which the curvature is exaggerated.

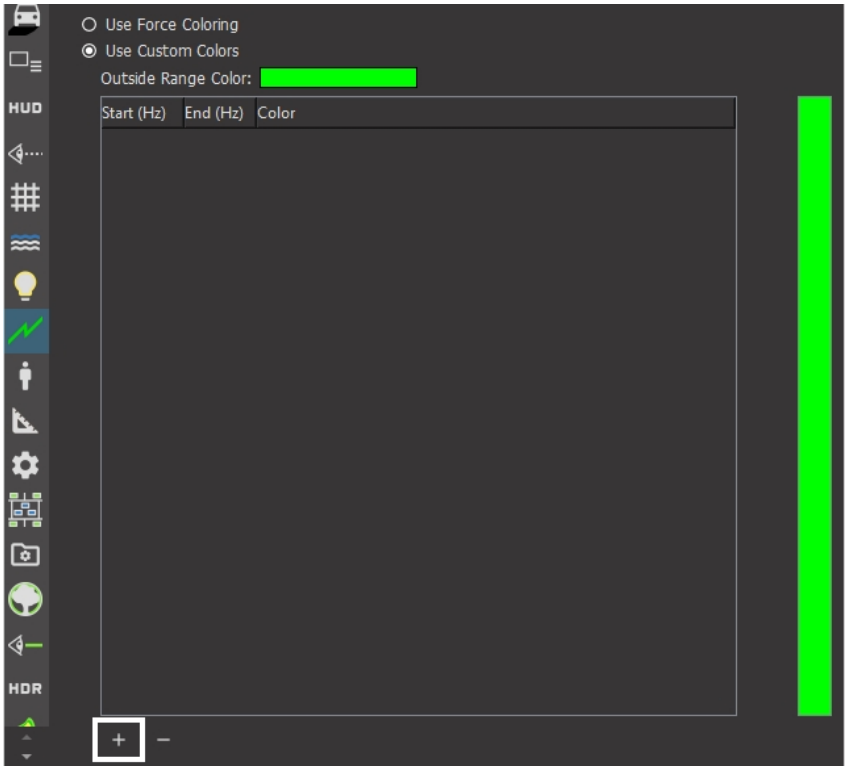
50.13.3 Configuring the Color of Radio Communication Lines

By default, radio communication lines are green. You can configure communication lines to use force coloring or to be colored based on the frequency of the message. When you color by frequency, you specify colors for the frequency ranges you are interested in and a color to use for all messages outside the specified frequencies.

➤ **To configure the color of radio communication lines:**

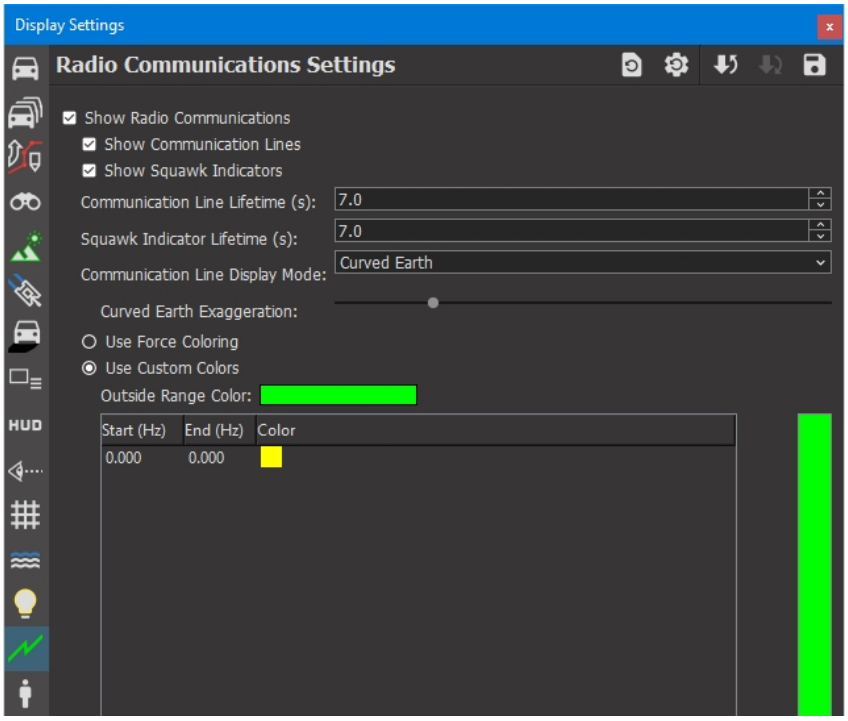
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Radio Communications Settings page (Figure 395 on the previous page).
3. To color radio communication lines using the force color, select the Use Force Coloring option.
4. To color radio communication lines by frequency:
    a. Select the Use Custom Colors option.
    b. Click the Add button (+). A range is added to the frequency list with a default color (Figure 396).

Figure 396. New frequency point



c. Double-click the starting value of the range to make it editable (Figure 397).

Figure 397. Editable frequency point



d. Type the starting frequency you want to use.

e. Edit the end frequency.

f. Double-click the color swatch and select a color for this range.

g. Optionally, add points for different frequency ranges.

h. Optionally, click the Outside Range Color color swatch and specify a color for all frequencies not specified in the frequency list.

If you configure multiple frequency ranges, you cannot add a range whose values are within those of an existing range.

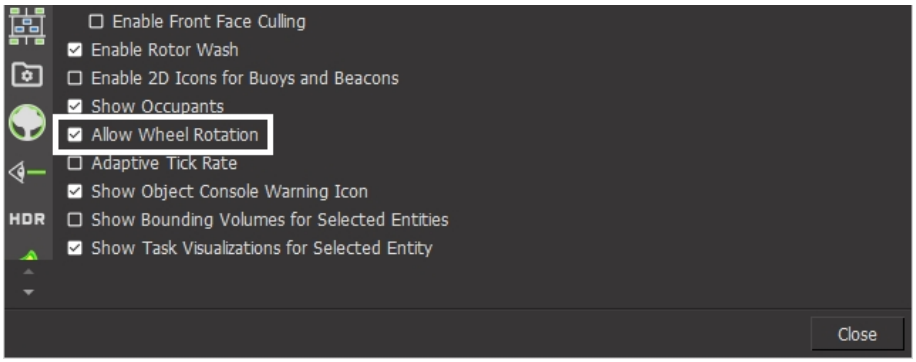
50.14. Enabling and Disabling Wheel Rotation

For vehicles that have articulated parts specified for the wheels, you can enable and disable rotation of the wheels. The rate of rotation is determined by the size of the wheel, the speed of the vehicle, and whether the vehicle is turning.

➤ **To enable or disable the rotation of vehicle wheels:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 398).

Figure 398. Wheel Rotation Settings



3. Select or clear the Allow Wheel Rotation check box.



51. Visualizing 2D Icons and 3D Models

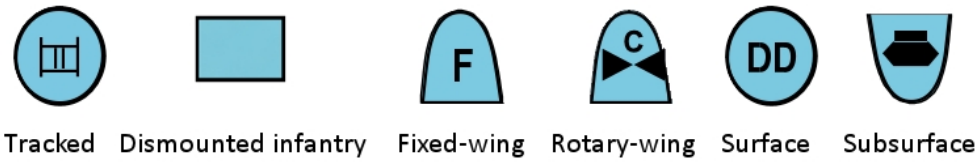
This chapter describes how to change how 2D icons and 3D models look.

51.1. Simulation Object Icons 1080
51.1.1 Changing the Size of 2D Icons 1080
51.1.2 Rotating 2D Icons to a Simulation Object's Heading 1082
51.2. Exaggerating the Scale of 3D Entity Models 1082
51.3. Hiding 3D Models 1084
51.4. Displaying DI-Guys in Vehicles 1085

51.1. Simulation Object Icons

The 2D Icons model set uses MIL-STD 2525B icons. Figure 399 illustrates basic icon shapes for friendly forces.

Figure 399. MILSTD-2525B icons



The color and shape code for 2D simulation object icons is:

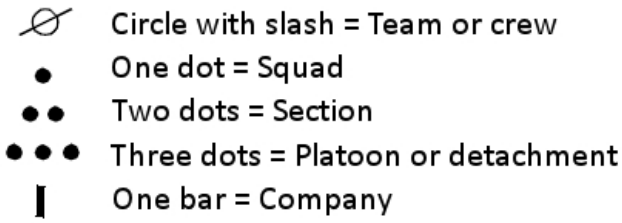
- Light blue – friendly.
- Pink (light red) – hostile.
- Light green – neutral.

Symbols have a black border except:

- If there is no border on the top, the entity domain is subsurface, for example, submarines.
- While selected, simulation objects have a white border.

Symbols that represent units have unit identifiers above the symbol, as illustrated in Figure 400.

Figure 400. Unit echelon indicators



51.1.1 Changing the Size of 2D Icons

The Plan View observer mode uses the 2D Icons model set for simulation objects.

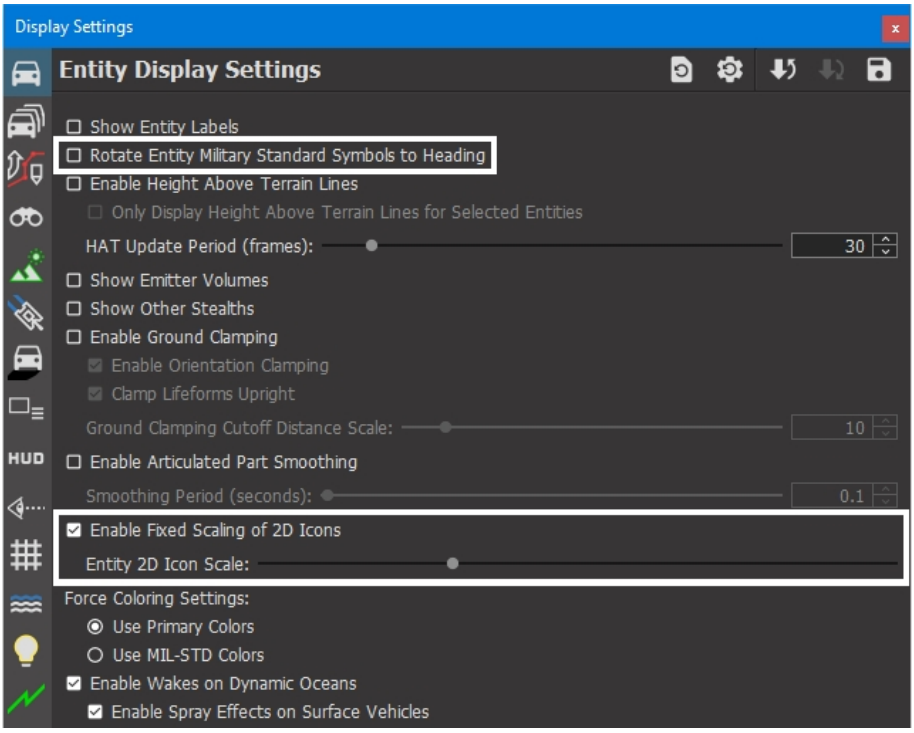
Manually Changing Icon Scale

You can change the size of 2D icons. When many simulation objects are close together, reducing the size of the icons can reduce overlap.

➤ **To change the size of simulation object icons:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 401).

Figure 401. Entity Display Settings page



3. Select the Enable Fixed Scaling of 2D Icons check box.
4. Adjust the Entity 2D Icon Scale slider.

Scaling 2D Icons by Altitude

Additionally, the 2D icons scale based on the observer's altitude, appearing smaller from greater heights. As with symbol decorations, the 2D icons display at full size when you hover over or select a simulation object.

➤ **To enable the resizing of icons based on altitude:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page.
3. Clear the Enable Fixed Scaling of 2D icons check box.

You can also enable or disable the resizing of icons based on altitude using the Display Settings Toolbar.

➤ To enable or disable the resizing of icons, click the Dynamically Scale Icons button .

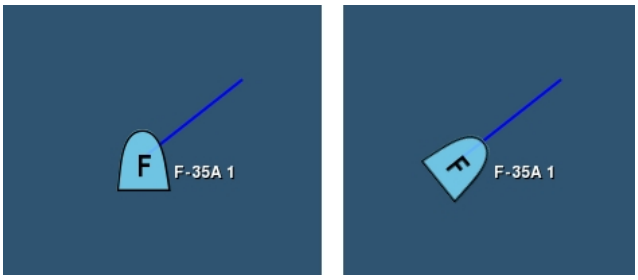
ⓘ Additionally, symbol decorations can be hidden based on the observer's altitude. See "21.2.3 Symbol Decorations" on page 473.

51.1.2 Rotating 2D Icons to a Simulation Object's Heading

By default, when a simulation object's heading changes, the heading indicator points to the new heading, but the icon remains oriented towards zero degrees (North). You can specify that icons rotate to the heading of the simulation object. This is enabled for airborne entities such as aircraft and munitions, but you can also enable it for other entities as needed.

> By default, this setting applies only to aircraft entities. If you want to override the Rotate Entity Military Standard Symbols to Heading setting for specific entities, such as ground entities, edit the Entity Definition and set the rotateicontoheading attribute for the Military Symbology Icon definition. See the chapter on *Element Definitions* in the *Adding Content to MAK ONE Applications Reference Manual*.

Figure 402. 2D icon rotated to heading



➤ **To rotate icons to a simulation object's heading:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 401 on the previous page).
3. Select the Rotate Entity Military Standard Symbols to Heading check box.

> You can also click the Rotate 2D MIL STD Symbols button ( ) in the Display Settings Toolbar.

51.2. Exaggerating the Scale of 3D Entity Models

If you view the terrain in Stealth observer mode, beyond a certain distance most 3D entity models are too small to see. Model scaling lets you exaggerate the scale of entity models to see them at a great distance. Figure 403 shows the same scene with model scaling off and on. When model scaling is off, only the largest entities are visible. When models are scaled, they are all visible.

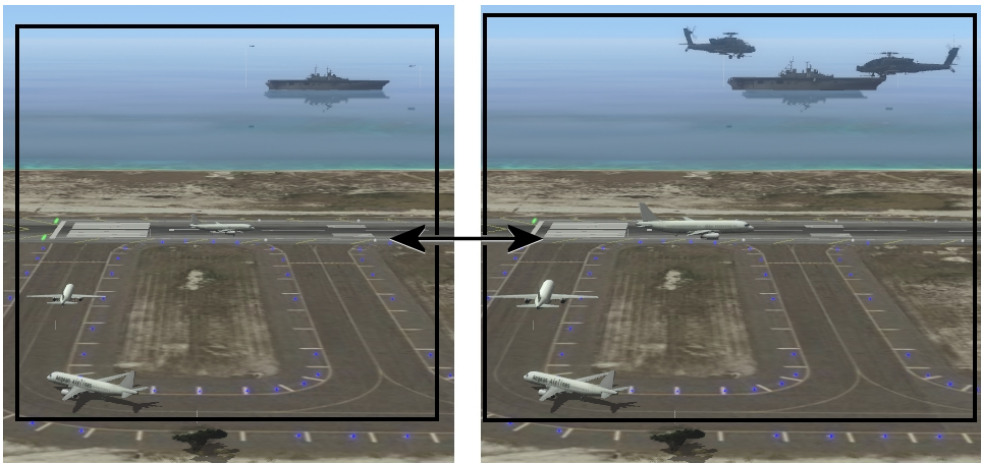
VR-Forces supports two methods of model scaling:

- XR scaling. XR scaling tries to give all entities approximately equal visibility. It does not preserve size relationships between entities. The visual effect of model scaling depends on an entity's size and the observer's distance from the entity. When the

observer is close to an entity, scaling models may not make any noticeable difference in its size.

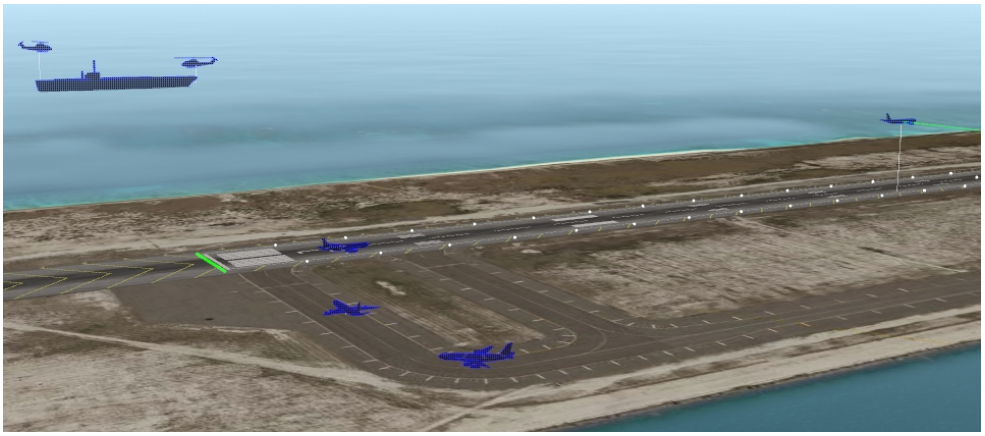
- Linear scaling. Linear scaling scales all entities by the current model scale factor. The distance from the observer is not taken into account. A model scale factor of 10 scales all entities to be 10 times normal size.

Figure 403. Model scaling off and on (Stealth observer mode)



Another way to quickly change the visibility of entities is to switch to XR observer mode. In XR mode, entities are immediately scaled and are represented by colorized 3D models to enhance their visibility.

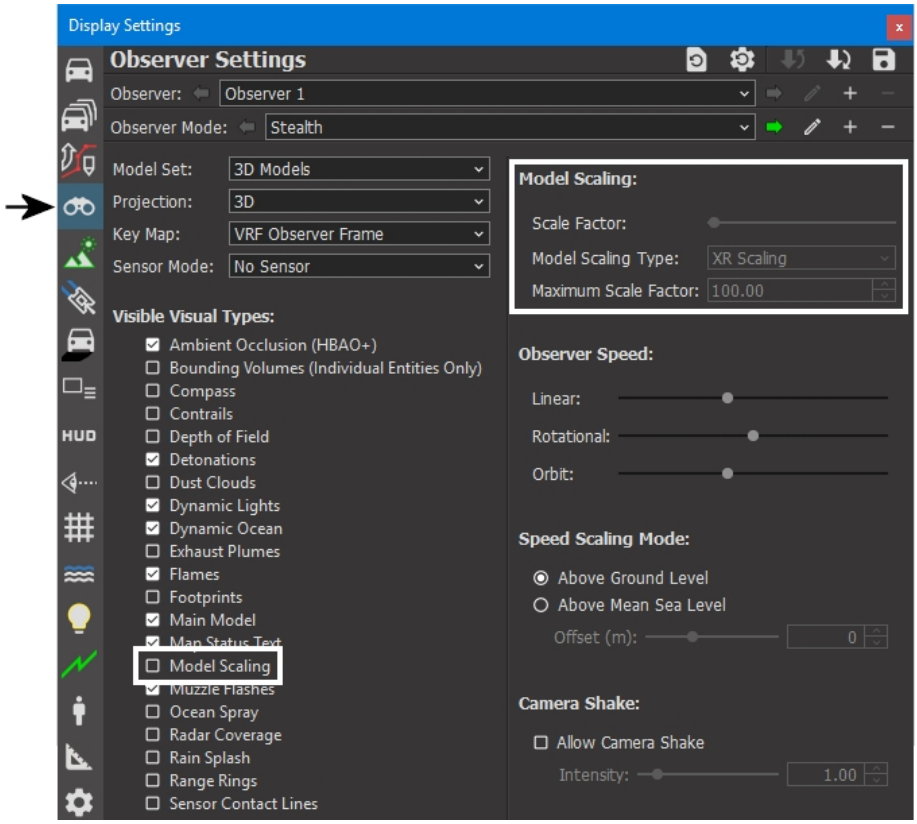
Figure 404. XR mode



➤ **To scale entity models:**

1. Choose **Observer > Advanced Settings**. The Display Settings dialog box opens to the Observer Settings page (Figure 405).

Figure 405. Observer Settings page



2. In the list of observer attributes, select Model Scaling.
3. In the Model Scaling Type list, select the type of scaling you want to use: XR or linear.
4. Optionally, change the Maximum Scale Factor.
5. Adjust the Scale Factor slider to the amount of scaling that you want.

51.3. Hiding 3D Models

You can hide 3D models and 3D colorized models.

➤ **To hide 3D models:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 405 above).
3. In the list of observer settings, clear the Main Model check box.

51.4. Displaying DI-Guys in Vehicles

If a vehicle has interior geometry, you can configure it to display DI-Guy characters inside the vehicle. If a vehicle is configured to display a DI-Guy, the model is added automatically. You can enable and disable display of DI-Guys. This is a global setting.

The characters displayed in vehicles are for display purposes only. They do not exist as entities in the simulation.

You can show and hide occupants on a per-entity basis with the Hide Occupants set data request. For details, see "40.40. Hide Occupants" on page 882.

➤ **To enable or disable display of DI-Guys in vehicles:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page.
3. Select or clear the Show Occupants check box.



52. Lighting Effects

This chapter describes the lighting effects supported by VR-Forces.

52.1. Displaying Lighting Effects 1088
52.2. Advanced Lighting Settings 1088
52.2.1 Configuring Ambient and Diffuse Lighting 1089
52.2.2 Specifying the Emissive Lighting Scale 1090
52.3. Displaying Dynamic Lighting 1090
52.3.1 Daylight Control of Lights 1092
52.4. Displaying Ocean Planar Reflections 1092
52.4.1 Configuring Ocean Planar Reflections 1093
52.5. Lens Flare 1094
52.6. Displaying Crepuscular (Sun) Rays 1095
52.7. Configuring the Skybox Cube Map 1096
52.8. Flashlight Mode 1097
52.9. Using High Dynamic Range Lighting 1097
52.10. Disabling Lighting and Clouds 1100
52.11. Displaying Shadows (3D Only) 1100
52.11.1 Enabling Shadows on Trees 1102
52.11.2 Configuring Shadows 1102

52.1. Displaying Lighting Effects

VR-Forces can simulate a variety of lighting effects, including:

- Dynamic lighting
- Ocean planar reflection
- Lens flare
- Crepuscular (sun) rays
- Shadows

Use of lighting effects affects performance. VR-Forces lets you enable and disable some effects so you can adjust the mixture of performance and effects that best meets your needs.

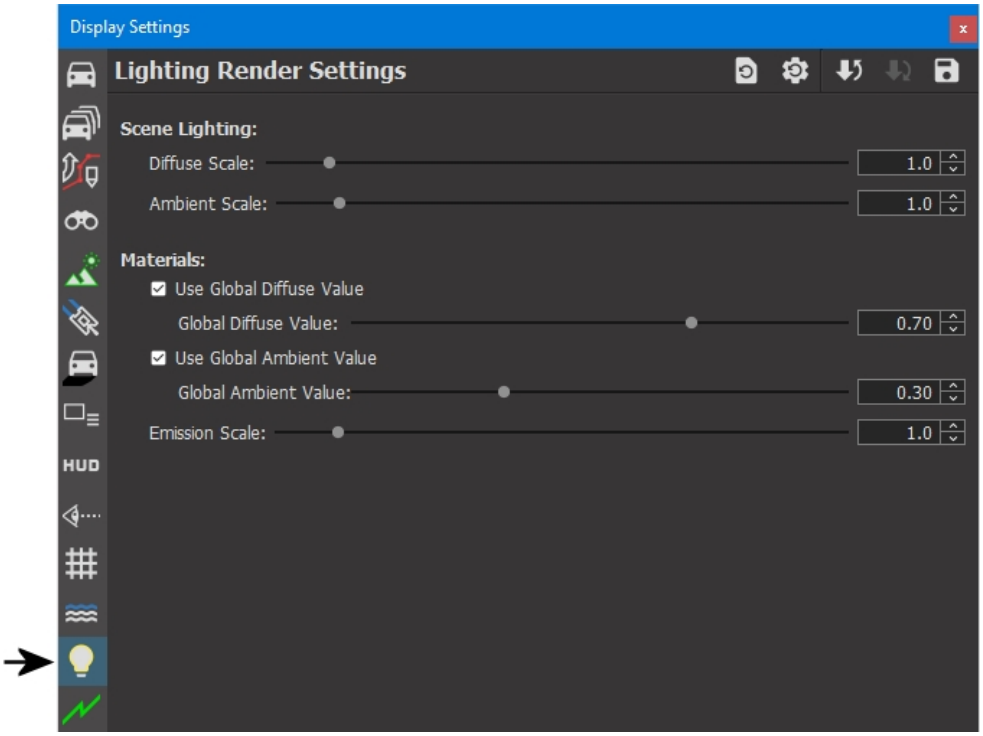
52.2. Advanced Lighting Settings

VR-Forces allows you to adjust the lighting for the scene and materials.

➤ **To access the lighting render settings:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 406).

Figure 406. Lighting Render Settings



52.2.1 Configuring Ambient and Diffuse Lighting

Ambient lighting is the lighting available to objects that are not in a direct form of light, such as sunlight. Diffuse lighting is light that is reflected off an object. You can adjust the overall amount of ambient and diffuse lighting in the scene and you can configure whether objects use the material ambient and diffuse values specified in their models or use values provided by VR-Forces.

Configuring the Ambient and Diffuse Scale Values for Scene Lighting

You can set a scale factor for ambient and diffuse light that is present in the scene. Lower values make objects darker. Higher values make them lighter. The ambient or diffuse scale factor is independent of how material ambient and diffuse values are set.

➤ **To specify the ambient and diffuse lighting scale values:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 406 on the previous page).
3. Adjust the Diffuse Scale slider or change the value in the box.
4. Adjust the Ambient Scale slider or change the value in the box.

Configuring the Material Ambient and Diffuse Values for Materials

Polygons in a 3D scene have a material and a texture. Polygon materials may specify specular, diffuse, and ambient colors that define the way they respond to incoming light.

The ambient color specifies the color used when it is in indirect light. Sometimes the ambient values for different objects do not work well together. You can specify that objects use the ambient material value, or let VR-Forces globally specify ambient material values.

In addition to specifying which material ambient and diffuse values to use, you can set a scale factor. Lower values make objects darker. Higher values make them lighter. The material ambient or diffuse scale factor is independent of how ambient and diffuse values are set for scene lighting.

➤ **To configure the material ambient value:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 406 on the previous page).
3. To use the ambient values specified by objects, clear the Use Global Ambient Value check box. To have VR-Forces use a global value, select the Use Global Ambient Value check box.
4. If you selected the Use Global Ambient Value check box, optionally, adjust the Global Ambient Value slider or change the value in the box.

➤ **To configure the material diffuse value:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 406 on page 1088).
3. To use the diffuse values specified by objects, clear the Use Global Diffuse Value check box. To have VR-Forces use a global value, select the Use Global Diffuse Value check box.
4. If you selected the Use Global Diffuse Value check box, optionally, adjust the Global Diffuse Value slider or change the value in the box.

52.2.2 Specifying the Emissive Lighting Scale

You can change a scale value that controls the brightness of emissive textures. This allows you to make windows, street lights, and other emissive light sources darker or brighter. Increasing the scale value increases the brightness.

➤ **To specify the emissive lighting scale value:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Lighting Render Settings page (Figure 406 on page 1088).
3. Adjust the Emission Scale slider or change the value in the box.

52.3. Displaying Dynamic Lighting

VR-Forces can display light cast by light sources such as headlights, street lamps, and spotlights. Display of dynamic lighting is an observer-specific feature. It is enabled by default.

Dynamic lights are either part of the terrain or part of an entity model. If they are part of an entity model, they can be set permanently on or they can be turned on and off by the appearance bit. VR-Forces entities that have lights can enable them using the Appearance set data request.

Figure 407 shows a night-time scene with dynamic lighting enabled. The truck's taillights cast light to the rear. The headlights cast light forward. Street lamps and windows are also casting light.

Figure 407. Dynamic Lighting enabled



- Light points can be configured to automatically turn on at dusk and turn off at dawn. For details, see "52.3.1 Daylight Control of Lights" on the next page.
- Emissive lights are always on, day or night. The headlights on vehicles shipped with MAK products are configured with Dynamic Lights and Emissive Textures which are controlled by the Appearance State "Head Lights On". When the headlights are on and dynamic lights are enabled, the emissive textures are visible.

You cannot change these lighting settings.

➤ **To enable dynamic lighting:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page.
3. Make sure the Observer Mode is set to Stealth, Out the Window, or another 3D view.
4. In the list of settings, select Dynamic Lights.

You can also enable or disable dynamic lighting by choosing **Observer > Dynamic Lights**.

52.3.1 Daylight Control of Lights

Lights that have a light point type specified can be placed under daylight control.

- Light points that do not have a light point type defined are not automatically placed under daylight control.
- Light points in an OpenFlight file that are loaded as an articulated model use the **daylightControlledLights** model definition parameter.
- Light points with a type defined use the Daylight Controlled parameter for their type in *./appData/importConfig/Lights_DefaultValues_3_2.csv*.

For more information about configuring light points, see "Configuring Light Points" in the *Adding Content to MAK ONE Applications Reference Manual*.

Light sources (lights that cast light onto other objects) are discovered on the cull pass of the scene graph. Sources found under the terrain root or props root of the scene graph are automatically put under daylight control.

52.4. Displaying Ocean Planar Reflections

VR-Forces can show the reflection of terrain and objects on water (Figure 408). To see planar reflections, dynamic ocean must be enabled. (For details about enabling dynamic ocean, see "11.8.1 Enabling Marine Effects" on page 342.)

Displaying ocean planar reflections degrades performance. See "52.4.1 Configuring Ocean Planar Reflections" on the facing page for details.

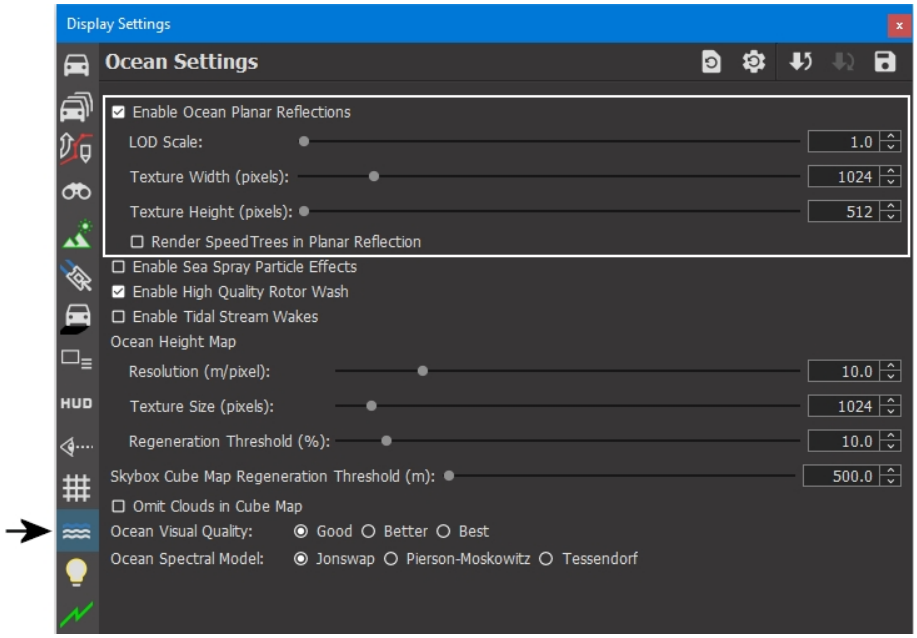
Figure 408. Ocean planar reflection



➤ **To enable or disable planar reflections:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 409).

Figure 409. Ocean Settings



3. Select or clear the Enable Ocean Planar Reflections check box.

52.4.1 Configuring Ocean Planar Reflections

As with other visuals, ocean planar reflections are controlled by the observer's LOD Distance Scale setting. To reduce the performance impact of ocean planar reflections, you can configure the LOD scale specifically for ocean planar reflections. The LOD scale affects the distance from the observer at which VR-Forces switches LODs. The higher the value, the more quickly LODs are switched.

> You can adjust the LOD Distance Scale for terrain in the Observer Control Panel (Figure 465 on page 1189).

The Texture Height and Texture Width parameters affect the resolution of the ocean planar reflection texture, in pixels. Higher resolution decreases performance.

You can also enable and disable display of SpeedTrees as planar reflections.

➤ **To configure ocean planar reflections:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 409 above).
3. To change the ocean planar reflection LOD scale, adjust the LOD Scale slider under Enable Ocean Planar Reflections, or change the value in the box.
4. To change the width or height of the ocean planar reflection texture, adjust the Texture Width or Texture Height slider under Enable Ocean Planar Reflections, or change the values in the boxes.

5. To include SpeedTrees in ocean planar reflections, select the Render SpeedTrees in Planar Reflections check box.

52.5. Lens Flare

Lens flare is a visual artifact often seen when looking at light sources such as the sun through a camera lens (Figure 410). VR-Forces can simulate lens flare when the observer looks towards the sun.

Figure 410. Lens flare

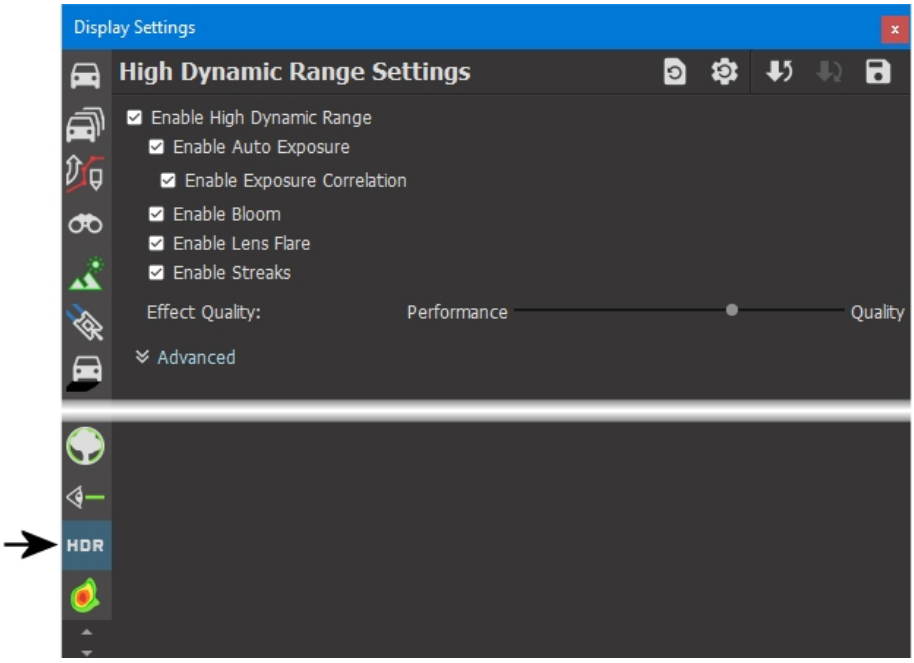


Lens flare is supported only when high dynamic range lighting is enabled. You can also enable this in the High Dynamic Range Settings page.

➤ **To enable or disable lens flare:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the High Dynamic Range Settings page (Figure 411).

Figure 411. High Dynamic Range Settings page

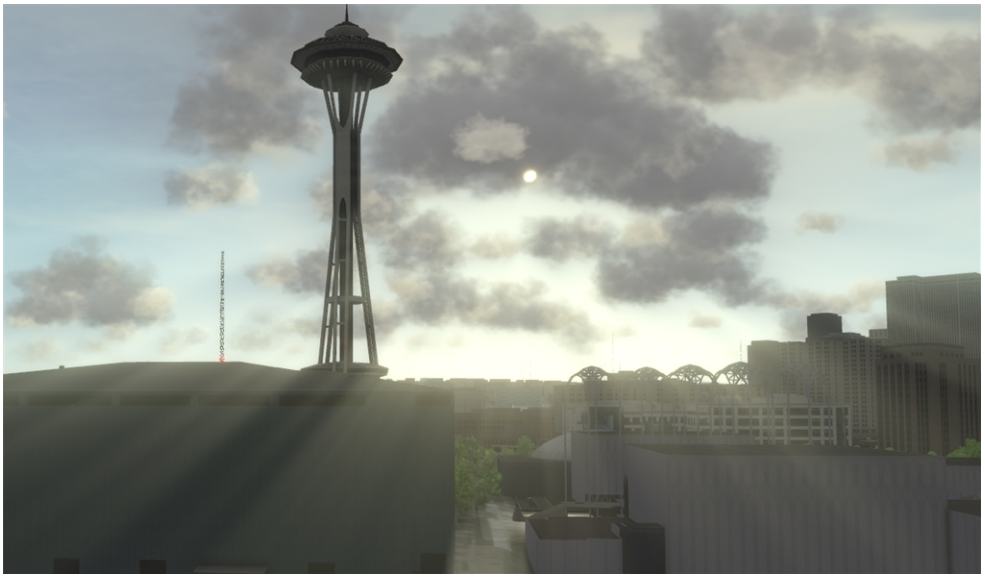


3. Select or clear the Enable Lens Flare check box.

52.6. Displaying Crepuscular (Sun) Rays

VR-Forces can simulate the effect of crepuscular rays (sunbeams) beaming through clouds (Figure 412).

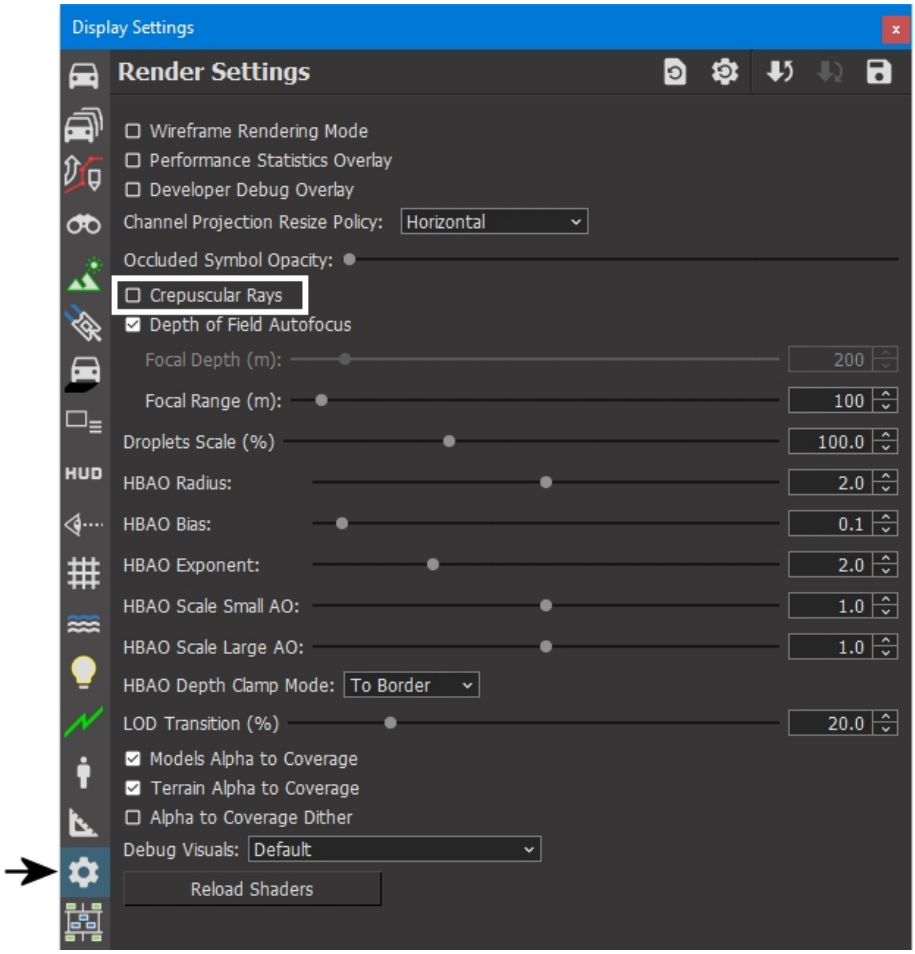
Figure 412. Crepuscular rays



➤ **To enable or disable crepuscular (sun) rays:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 413).

Figure 413. Render Settings page



3. Select or clear the Crepuscular Rays check box.

52.7. Configuring the Skybox Cube Map

A skybox cube map is a graphics technique used to render reflections and sky effects. The skybox cube map is rendered from the location of the observer. As the observer moves, the cube map needs to be recalculated, which is computationally expensive.

VR-Forces lets you configure the distance that the observer has to move (the skybox cube map regeneration threshold) before it recalculates the cube map. Increasing the threshold can improve performance, at the cost of visual quality.

You can also choose to omit clouds in the skybox cube map. Omitting clouds can increase performance.

➤ **To configure the skybox cube map:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 409 on page 1093).
3. Adjust the Skybox Cube Map Regeneration Threshold slider or enter a value in the box.
4. Optionally, select the Omit Clouds in Cube Map check box.

52.8. Flashlight Mode

VR-Forces can light a circular area as if the observer is using a aflashlight. You can move the flashlight beam and increase or decrease the diameter of the flashlight beam.

Figure 414. Flashlight mode



➤ To turn flashlight mode on or off, press **F**.

➤ To move the flashlight beam, hold down **Alt** and the right mouse button, then move the mouse.

➤ To change the size of the flashlight beam, hold down **Alt** and move the mouse scroll wheel forward or back.

52.9. Using High Dynamic Range Lighting

High dynamic range (HDR) lighting makes scenes look more realistic. They look particularly good on HDR monitors. HDR is enabled by default.

➤ **To configure HDR lighting:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the High Dynamic Range Settings page (Figure 411 on page 1095).
3. To enable or disable HDR lighting, select or clear the Enable High Dynamic Range check box.
4. Enable other settings as desired.
5. For those settings that are enabled, adjust their parameters. Table 44 describes the HDR settings.

Table 44. HDR settings

Setting	Description
Enable High Dynamic Range	Turns High Dynamic Range (HDR) on and off. When HDR is turned on, the scene is rendered in 16 bits per color floating point and then tone mapped to 8 bits per color before being sent to the display. Additionally, there are post-process effects applied to objects that are greater than the display range.
Enable Auto Exposure	Turns auto exposure on and off, which is a post process that scales the intensity of the rendered image based on the intensity of the screen. When auto exposure is turned off, the global lighting values are used to scale the intensity to a reasonable value.
Enable Exposure Correlation	When auto exposure is enabled, this controls whether all the channels share the same auto exposure adjustment. When enabled, all channels use the exposure for the main luminance. If if it disabled, each channel across master or slave computes its own luminance.
Enable Bloom	Turns the bloom post process effect on and off. This effect is a blurring of high intensity objects in the scene which gives a glow around bright objects like light points.
Enable Lens Flare	Turns the lens flare post process effect on and off. This effect creates a lens flare effect for high intensity objects.
Enable Streaks	Turns the streak post process effect on and off. This effect creates a star effect by streaking the high intensity objects in all directions.
Advanced Settings	
Dynamic Light Scale	When HDR is turned on, the intensity of dynamic lights are scaled by this setting.
Emissive Texture Scale	When HDR is turned on, the intensity of the emissive textures is scaled by this setting.
Light Point Scale	When HDR is turned on, light point intensity is scaled by this setting.

Table 44. HDR settings (continued)

Setting	Description
Shininess Scale	When HDR is turned on the lighting shininess component is scaled by this setting. Exposure adjustments are all related to how the intensity of the scene is scaled before being tone mapped. These values are in order based on when they are applied to the scalar.
Specular Scale	When HDR is turned on the lighting specular component is scaled by this setting.
Rate Scale	When Auto Exposure is enabled, this scales the time it takes for the auto exposure to adjust to changes in the scene. Increase this value to reduce the time it takes for the auto exposure to adjust.
Pre Scale	When Auto Exposure is enabled, this scales the calculated intensity for the scene. Increase this value to reduce the overall intensity of the final image.
Min Difference	When Auto Exposure is enabled, this clamps the minimum value of the calculated intensity to a portion of the global lighting value. This is used to prevent the auto exposure from going too low compared to the global lighting value.
Max Difference	When Auto Exposure is enabled, this clamps the maximum value of the calculated intensity to a portion of the global lighting value. This is used to prevent the auto exposure from going too high compared to the global lighting value.
Post Scale	Scales the scalar after all Auto Exposure calculations have been done. This also affects the global lighting value when Auto Exposure is turned off. Increase this value to reduce the overall intensity of the final image.
Final Clamp Min	The minimum value for the final value that will be used to scale the intensity prior to tone mapping.
Final Clamp Max	The maximum value for the final value that will be used to scale the intensity prior to tone mapping.
Bloom Scale	Scales the bloom before it is applied to the scene. Increase this value to increase the intensity of the bloom effect on high intensity objects.
Lens Flare Scale	Scales the lens flare effect before it is applied to the scene. Increase this value to increase the intensity of the lens flare effect.
Streak Intensity Scale	Scales the streaks before they are applied to the scene. Increase this value to increase the intensity of the streak effect on high intensity objects.
Number of Streaks	Number of streaks that the streak effect will create for each high intensity object. Maximum of 8 streaks.
Streak Length Scale	Scales the length of the streak effect. Increase this value to increase the length of the streak effect.

Table 44. HDR settings (continued)

Setting	Description
Streak Rotation Offset	Rotates the streak effect so that it changes the direction that the streaks point.
Gamma	Final color correction used to adjust the image based on the display. Change this value based on the characteristics of the display being used.

52.10. Disabling Lighting and Clouds

When you are using XR observer mode and you move the observer out from the terrain, the lighting and cloud effects can interfere with your ability to view simulation objects. Therefore, VR-Forces lets you quickly disable lighting effects and clouds. By default, they are enabled in Stealth observer mode and disabled in XR observer mode.

➤ To enable or disable lighting and clouds, choose **Observer > Sky**.

52.11. Displaying Shadows (3D Only)

VR-Forces can display shadows for props, entities, and terrain features (Figure 415). It can also display shadows on the ocean.

Figure 415. Shadows



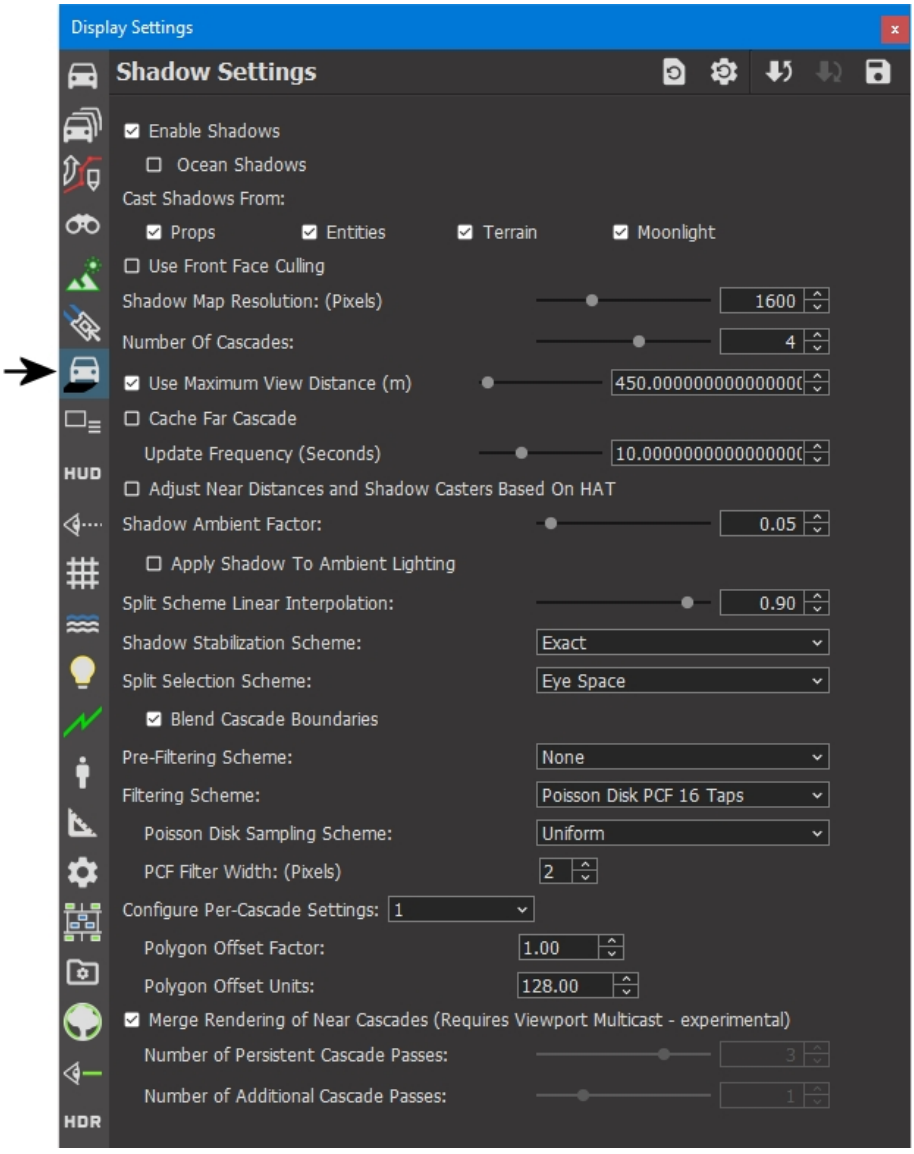
You can enable and disable shadows, and you can also configure shadow quality.

Enabling shadows can affect performance.

➤ **To enable and disable shadows globally:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Shadow Settings page (Figure 416).

Figure 416. Display Settings, Shadow Settings page



3. Select or clear the Enable Shadows check box to controls shadows on the terrain.
4. Select or clear the Ocean Shadows check box to control shadows on only the dynamic ocean.
5. To enable or disable shadows for specific object types, select or clear the check boxes under Cast Shadows From.

ⓘ You can also enable and disable shadows by choosing **Settings > Shadows**.

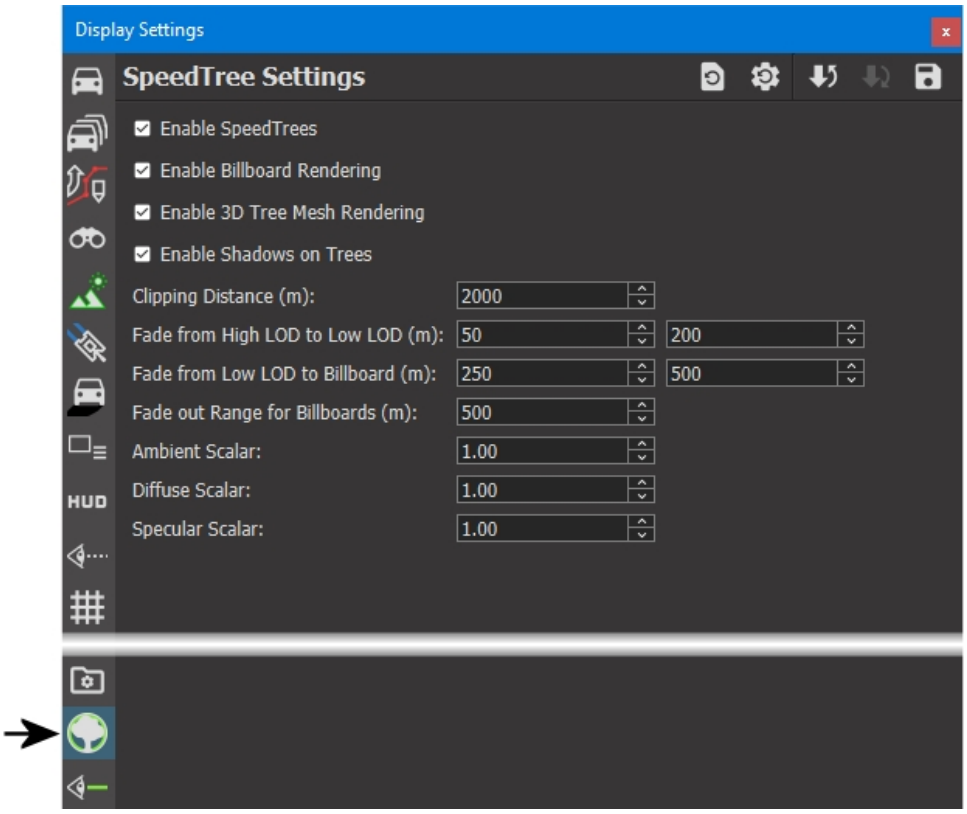
52.11.1 Enabling Shadows on Trees

You can enable and disable casting of shadows on SpeedTree trees.

➤ **To enable or disable casting shadows on SpeedTree trees:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the SpeedTree Settings page (Figure 417).

Figure 417. SpeedTree Settings page



3. Select or clear the Enable Shadows on Trees check box.

52.11.2 Configuring Shadows

You can configure which objects cast shadows and parameters that affect performance and the quality of shadows.

Shadows are drawn based on a shadow map. A shadow map determines what parts of a scene are in shadow relative to a light source. Then the scene is drawn showing the shadows that would be visible from the viewpoint of the observer.

Scenes are split into sections, called cascades. The cascades that are closest to the camera show the highest resolution shadows. Many of the settings control the size of cascades and how they merge together.

For an in-depth understanding of how shadows are rendered in 3D scenes, please see the many articles available on the world wide web.

➤ **To configure how shadows are rendered:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Shadow Settings page (Figure 416 on page 1101).
3. Change the parameters as desired. Table 45 briefly describes them.

Table 45. Shadow settings

Setting	Description
Use Front Face Culling	Use of face front culling is a choice between more shadow acne (an effect similar to Z-fighting) and more light bleeding artifacts.
Shadow Map Resolution	Increasing shadow map resolution provides more detail at the cost of performance and increased memory use.
Number of Cascades	The more cascades there are, the more detail is rendered, at the expense of memory usage and performance.
Use Maximum View Distance	If enabled, specifies how far away shadows are displayed. The smaller the area in which shadows are displayed, the more quickly they are displayed and the higher the quality. If disabled, uses the scene values.
Cache Far Cascade	Caches the farthest cascade and updates it less frequently than the near cascade, which increases performance.
Adjust Near Distances and Shadow Casters Based On HAT	As the observer moves higher above the terrain, the near cascade shrinks and only entities are rendered to them. The far cascade becomes responsible for the terrain shadows.
Shadow Ambient Factor	Fades out shadows. It determines how much light gets through.
Apply Shadow to Ambient Lighting	By default, shadows only affect the diffuse and specular lighting. Enable this option to also apply shadows to ambient lighting. This can make shadows darker.
Split Scheme Linear Interpolation	Adjusts how cascades are distributed between a logarithmic approach and a linear approach.
Shadow Stabilization Scheme	A list of different ways to reduce the shimmer that is common in shadow maps.
Split Selection Scheme	A list of different ways to choose which cascade a piece of geometry is in.
Pre-Filtering Scheme	A list of different shadow casing techniques to try to smooth shadows.
Filtering Scheme	Affects how blurred the shadow map is.

Table 45. Shadow settings (continued)

Setting	Description
Configure Per-Cascade Settings	Lets you configure the polygon offset for cascades.
Merge Rendering of Near Cascades	Lets you draw cascades all at once instead of individually. This improves performance. It is available on supported video cards (NVIDIA 980 or newer). If disabled, you can configure the number of cascades passes to draw shadows.



53. Marine Effects

This chapter describes different marine effects and water droplets.

53.1. Displaying Water Droplets and Splash Effects 1106
53.1.1 Enabling Sea Spray 1107
53.1.2 Setting the Size of Water Droplets 1108
53.1.3 Enabling and Disabling Rotor Wash 1109
53.1.4 Enabling High Quality Rotor Wash 1110
53.2. Displaying Wakes and Spray Effects 1110
53.3. Enabling Buoyancy for Surface Entities 1111

53.1. Displaying Water Droplets and Splash Effects

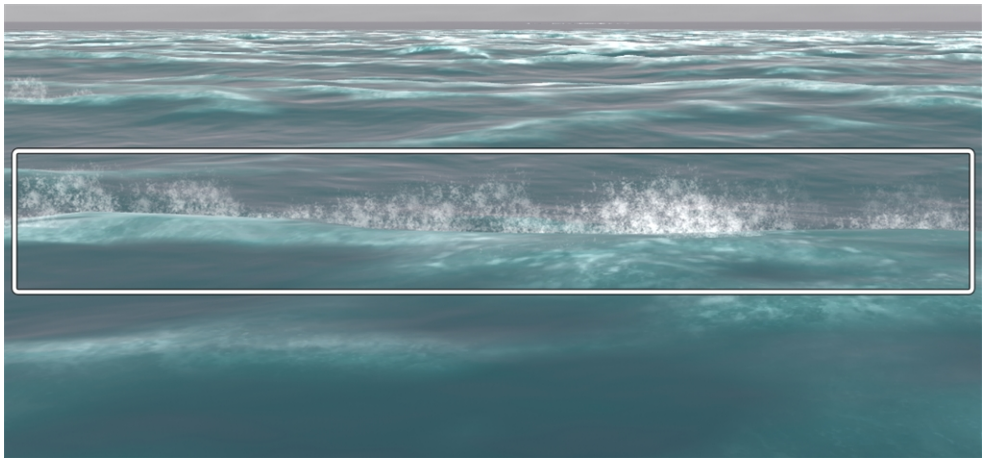
When it is raining, VR-Forces can simulate raindrops hitting the observer's camera lens (Figure 418). If the observer is close to the surface of the ocean (as if a periscope were being raised or lowered), VR-Forces can simulate the effect of waves and water droplets splashing on the observer's camera.

Figure 418. Rain effect



When dynamic ocean is enabled, VR-Forces can simulate sea spray (most visible in high seas) (Figure 419). It can also display rotor wash for hovering helicopters. You can display rotor wash at two levels of quality.

Figure 419. Sea spray effects

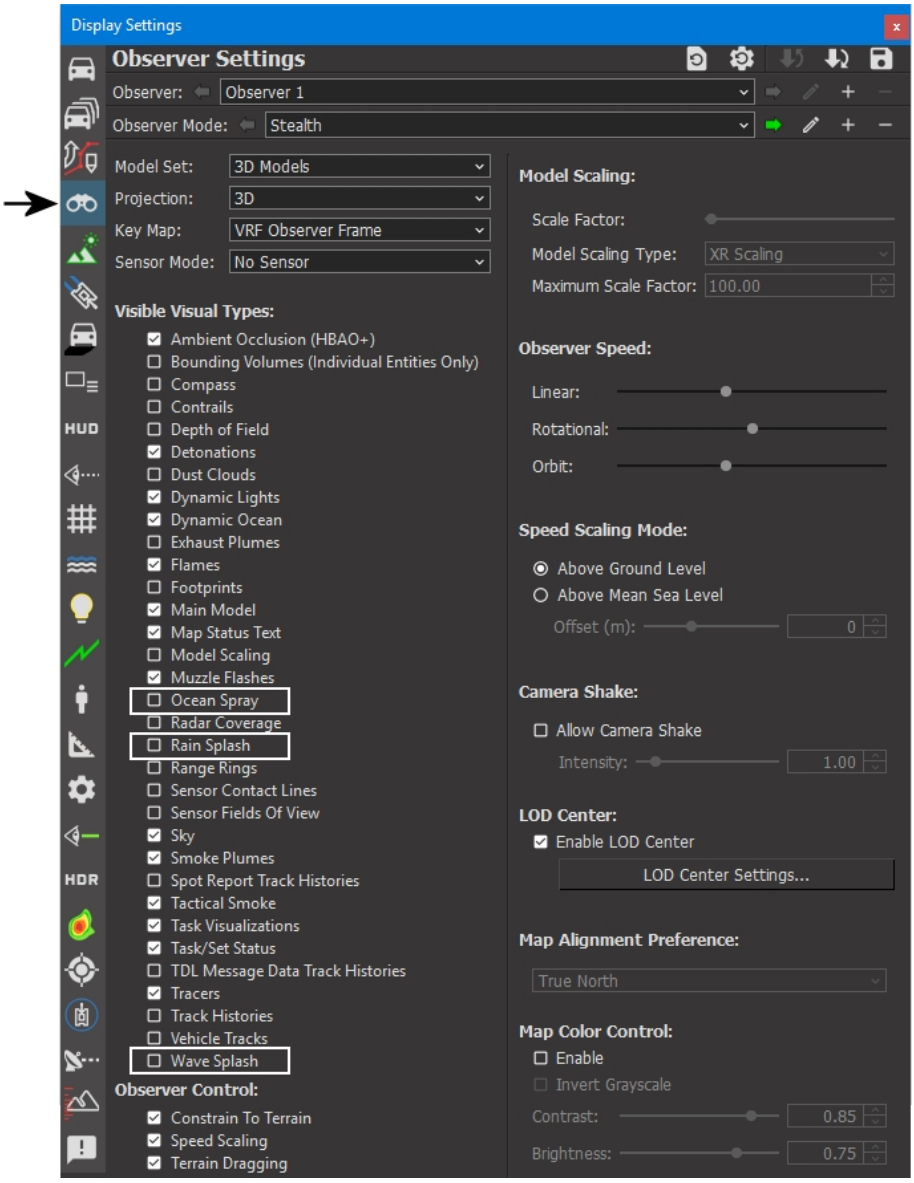


You can change the size of the water droplets for splash effects and sea spray. For details, see "53.1.2 Setting the Size of Water Droplets" on page 1108.

➤ **To enable or disable splash effects on the camera:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 420).

Figure 420. Observer Settings page



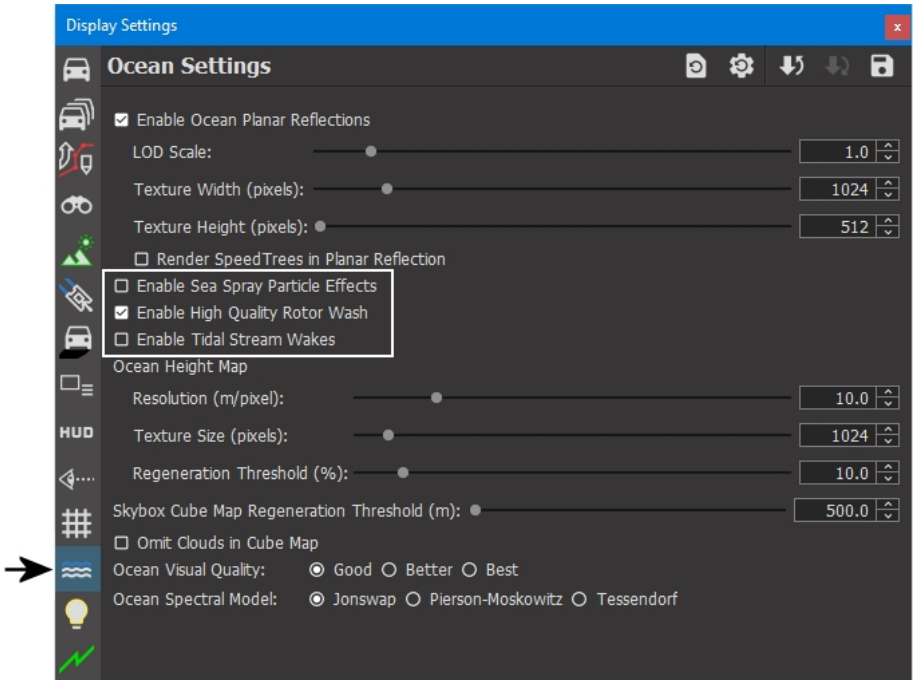
3. Select or clear the Ocean Spray, Rain Splash, or Wave Splash visible visual type check boxes.

53.1.1 Enabling Sea Spray

➤ **To enable sea spray effects:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 421).

Figure 421. Ocean Settings page



3. Select Enable Sea Spray Particle Effects.

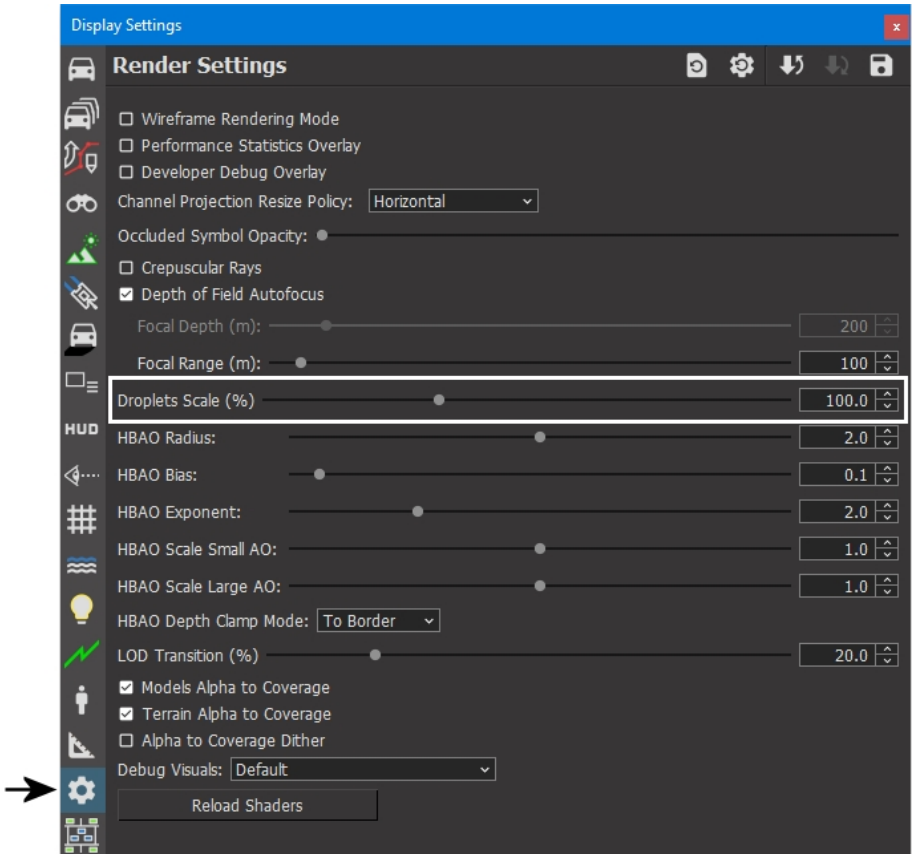
53.1.2 Setting the Size of Water Droplets

You can change the size of water droplets for splash effects and sea spray.

➤ **To change the size of water droplets:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 422).

Figure 422. Render Settings page



3. Adjust the Droplets Scale slider or enter a value in the box.

53.1.3 Enabling and Disabling Rotor Wash

If dynamic ocean is enabled, VR-Forces can display rotor wash over the water for rotary-wing entities. VR-Forces can also display rotor wash over land for some soil types. You can enable or disable rotor wash globally.

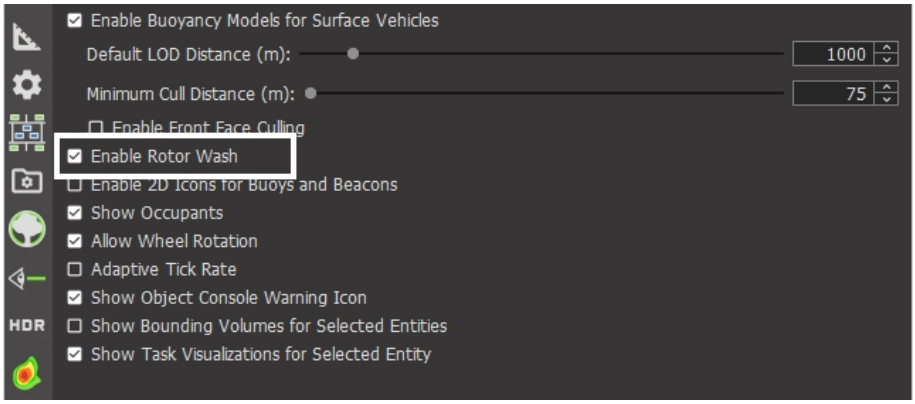
Rotor wash is a particle system. You can edit rotor wash particle systems in the Particle System Editor. For details, see *Special Effects (Particle Systems)* in the *Adding Content to MAK ONE Applications Reference Manual*.

> VR-Forces displays rotor wash only for helicopters that are simulated by VR-Forces. There is no DIS or HLA standard for rotor wash. VR-Forces sends special messages to enable its display.

➤ **To enable or disable rotor wash:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 423).

Figure 423. Rotor wash option



3. Select or clear the Enable Rotor Wash check box.

53.1.4 Enabling High Quality Rotor Wash

➤ **To enable or disable high quality rotor wash:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Ocean Settings page (Figure 421 on page 1108).
3. Select or clear the Enable High Quality Rotor Wash check box.

53.2. Displaying Wakes and Spray Effects

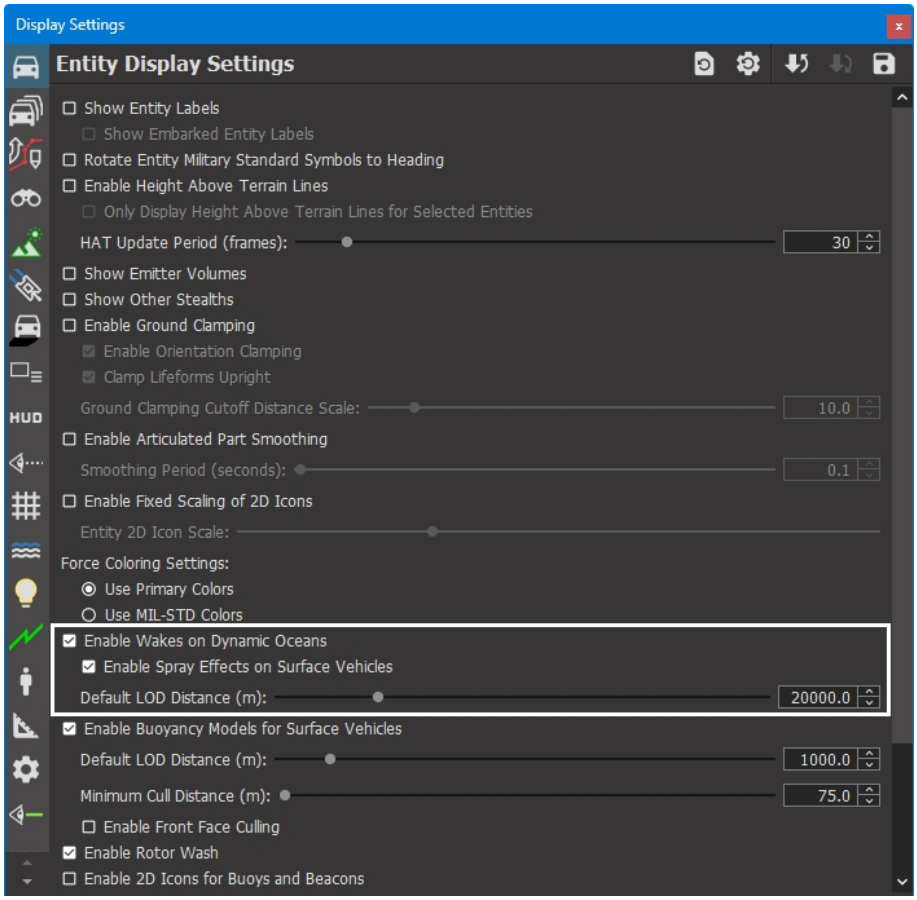
When dynamic ocean is enabled, surface entities display wakes and spray effects. However, these effects affect performance. You can disable them if you need to emphasize performance over visual effects.

Wakes have their own schema, with many attributes, including length, offset, the amount of time over which they fade out, and LOD distance. You can edit these attributes for individual entities in the Visual Model Editors dialog box.

➤ **To enable or disable wakes and spray effects:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 424).

Figure 424. Entity Display Settings page



3. To enable or disable wakes, select or clear the Enable Wakes on Dynamic Oceans check box.
4. If wakes are enabled, select Enable Spray Effects on Surface Vehicles to enable spray effects. Clear the check box to disable them. Disabling wakes also disables spray effects.

53.3. Enabling Buoyancy for Surface Entities

When dynamic ocean is enabled, surface entities can bob up and down with wave action. Be aware that the buoyancy models affect performance. You can disable buoyancy if you need to emphasize performance over visual effects. You can also control the area where buoyancy models are calculated.

If buoyancy is enabled, VR-Forces calculates buoyancy only for models within certain distances of the observer:

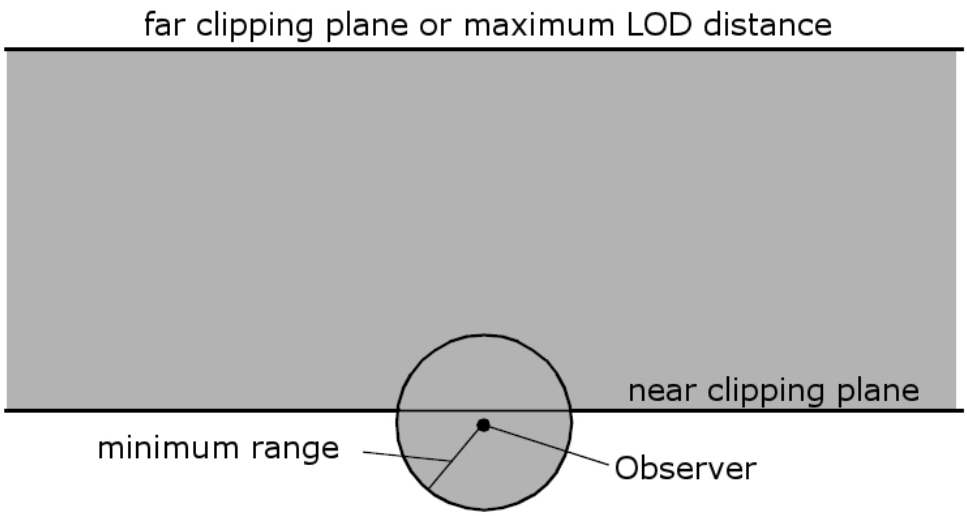
- Buoyancy models are always calculated for entities within the minimum cull distance. This ensures that if the observer turns, entities that are relatively near show buoyancy immediately.

- Buoyancy models are not calculated for entities that are beyond the maximum buoyancy LOD range. You can set this value on a per-entity basis in its entity model definition. If it is not set in the model definition, VR-Forces uses the default LOD distance specified on the Entity Display Settings page. If the default distance is set to 0.0, then entities that do not have an LOD range set in their model definitions have an infinite buoyancy range (although buoyancy is still not calculated past the far clipping plane).
- Buoyancy models are not calculated for entities beyond the far clipping plane or nearer than the near clipping plane.

VR-Forces uses the nearer of the far clipping plane or the maximum buoyancy LOD range to determine the distance at which it stops calculating buoyancy models.

When buoyancy is enabled, you can specify that VR-Forces use face front culling. VR-Forces calculates buoyancy models for all entities within the minimum range and all entities between the near clipping plane and the far clipping plane or maximum buoyancy LOD range, whichever is closest. In Figure 425, the shaded area is the area in which buoyancy models are calculated.

Figure 425. Face front culling



By default, face front culling is disabled and only distance culling is used.

➤ **To enable or disable the buoyancy models:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 424 on the previous page).
3. Select or clear the Enable Buoyancy Models for Surface Vehicles check box.
4. To specify the distance beyond which buoyancy models are not calculated, adjust the adjacent Default LOD Distance slider or change the value in the box. (If the default distance is set to 0, then entities that do not have an LOD range set in their model definitions have an infinite buoyancy range. However, buoyancy is not

calculated past the far clipping plane.)

5. To specify the radius within which all entities have their buoyancy models calculated, adjust the Minimum Cull Distance slider or change the value in the box.
6. Optionally, specify use of face front culling.



54. Using CameraFX Sensor Views

This chapter explains how to enable and configure simulated night-vision goggle (NVG) and infrared views of the scene.

54.1. Using Sensors 1116
54.2. Enabling Sensors 1116
54.3. Configuring Sensors 1117
54.4. Adding New Sensors 1119

54.1. Using Sensors

VR-Forces can simulate the way that the 3D scene would look if an observer were using visual sensor devices such as night vision goggles or viewing the infrared spectrum (Figure 426). This is an observer-specific setting. You can adjust the contrast, blur, and noise of the view.

VR-Forces includes one sensor module: CameraFX, which uses the SenSim libraries from JRM Technologies. The sensors do not take into account the materials of the objects that you are viewing. They simply filter the view to produce the desired effect. CameraFX supports only one "ActiveRegion" per entity based on the power plant status. If you want physics-based sensor effects, you can purchase the SensorFX Plug-in for VR-Forces, developed by JRM. For details, please contact your MAK salesperson.

Figure 426. CameraFX - dawn, no sensor and IR sensor



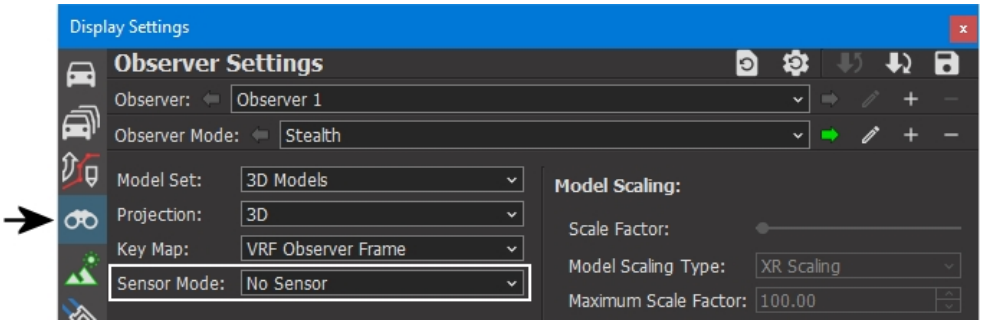
The sensors described in this chapter are strictly visual effects. They have no relationship to the sensor systems used by simulation objects. They may not accurately reflect hotspots such as humans or vehicle engines.

54.2. Enabling Sensors

Sensors are observer-specific (Figure 427). VR-Forces includes observer modes that have sensors enabled. The names of the observer modes match the sensor modes that they enable.

- NVG (CameraFX) (night vision goggles).
- IR (CameraFX) (infrared).
- IR BlackHot (CameraFX).
- Camera (CameraFX). This mode uses the Visual Camera sensor mode.

Figure 427. Sensor setting for observer mode



The default settings for the Camera observer mode make it look very similar to Stealth observer mode. The difference is that when you use the Camera observer mode, you can change the view by changing the sensor settings, such as blur, noise, and gain.

You can also specify use of a sensor configuration as a channel attribute.

➤ **To enable sensors, do one of the following:**

- Select an observer mode that has a sensor configured.
- Edit an observer mode to use a sensor.
- Create a new observer mode and configure it with a sensor.
- Change the Sensor attribute for a channel.

For details about creating and editing observer modes, see "59.4. Creating and Editing Observer Modes" on page 1166. For details about editing channel attributes, see "83.4. Changing a Channel's Attributes" on page 1604.

54.3. Configuring Sensors

You can configure the following aspects of sensors:

- Blur. The sharpness of the image.
- Noise. The graininess of the image.
- Gain. The contrast and brightness of the image. (IR only)
- Lighting overrides. You can adjust the amount of ambient and diffuse light and the intensity of light emitted by DI-Guys and other entities.
- Sensor texture. VR-Vantage supports a shader-based effects map called a Sensor Texture map. This is a three-channel texture that adds the red, green, or blue channels as appropriate when the observer is in an NVG CameraFX or IR mode. You can enable and disable use of these textures and specify their intensity. For more information about shader-based effects maps, see "Using Shader-based Effect Maps" in the *Adding Content to MAK ONE Applications Reference Manual*.

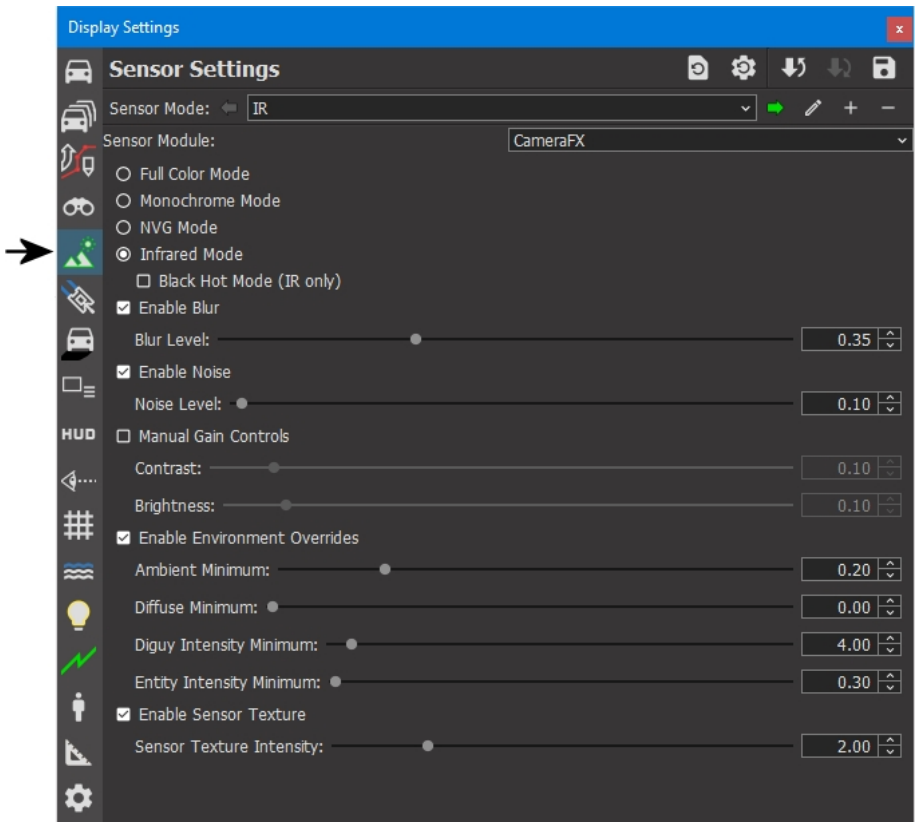
You can configure a separate set of settings for each sensor type. VR-Forces remembers the settings. When you change the settings for a sensor type, they affect the display only if that sensor type is in use. For example, if you select NVG in the sensor list on the Observer

Settings page, and select IR in the Sensors list on the Sensor Settings page, the changes you make to the IR configuration do not affect the NVG display you are looking at. If you then switch to IR in the VR-Forces window, the new IR settings take effect.

➤ **To configure sensors:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page (Figure 428).

Figure 428. Sensor Settings page



3. In the Sensor Mode list, select the sensor that you want to configure.
4. Select the Sensor module from the list.
5. Select the sensor mode (Monochrome, NVG, and so on). If you select Infrared Mode, select or clear the Black Hot Mode check box.

- It may seem obvious that if, for example, you select NVG from the Sensor Mode list, you would select it for the sensor type. However, you can add additional sensor configurations to the list and, in that case, you would need to specify the correct sensor type for the new sensor. For example, you might create a sensor called NVG-High Contrast. Then you would select NVG as the sensor type and adjust the contrast accordingly.
- Full Color Mode provides natural coloring, as you would see with no sensor modes applied. The benefit of creating a sensor mode using this option is that you can apply noise or blur to the scene.
- Monochrome Mode simply simulates a black-and-white display, such as closed-circuit television monitors. It is not the same as Infrared Mode, which provides a basic (not physics-based) simulation of either white- or black-hot infrared.

6. For each setting you want to change, select the controlling check box and adjust the sliders.

54.4. Adding New Sensors

You can add, rename, and delete sensors.

➤ **To add a sensor:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Sensor Settings page (Figure 428 on the previous page).
3. On the Sensor Mode line, click the Add button (+). A new sensor called sensor_*number* is added. There are no configuration options on the page. (To rename the sensor, click the Rename button (✎) and enter a new name.)
4. In the Sensor Module list, select CameraFX. The available configuration options are added to the page.
5. Select the type of sensor that you want this to be (Full Color, Monochrome, Infrared, or NVG).
6. Set the Blur, Noise, and Gain levels.



55. Intervisibility and Range

This chapter explains how to use the intervisibility and range line features.

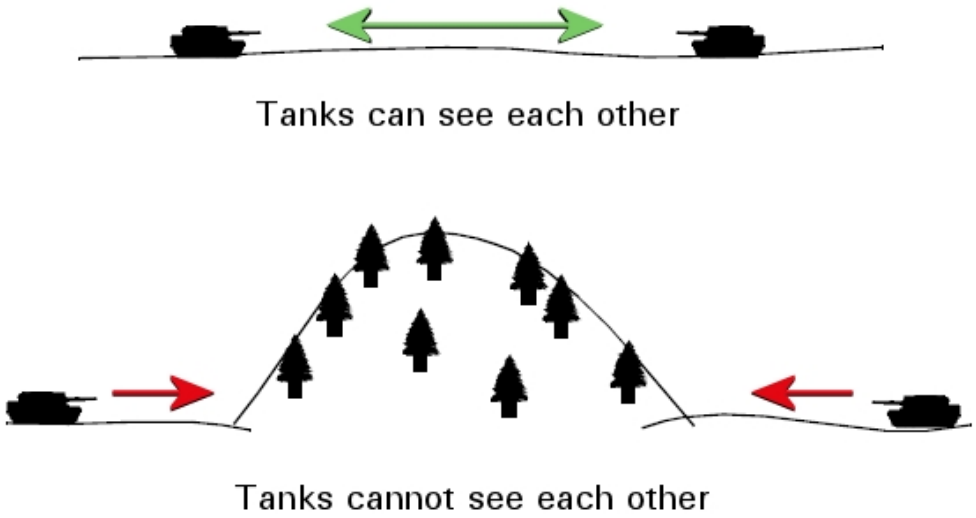
55.1. Intervisibility (Line-of-Sight) 1122
55.2. Types of Intervisibility 1123
55.2.1 Point-to-Point Intervisibility 1123
55.2.2 Intervisibility Fan 1123
55.2.3 Simulation Object Intervisibility 1124
55.3. Intervisibility Objects Can Be Transient or Permanent 1124
55.3.1 Creating a Transient Intervisibility Object 1125
55.3.2 Creating a Permanent Intervisibility Object 1126
55.4. Editing an Intervisibility Object 1126
55.5. Configuring Intervisibility Lines and Fans 1127
55.6. Pinning an Intervisibility Object to a Simulation Object 1128
55.7. Showing and Hiding Intervisibility Lines 1129
55.8. Displaying an Intervisibility Line's Terrain Profile 1129
55.9. Deleting Intervisibility Objects 1130
55.10. Range Lines 1130
55.10.1 Creating a Range Line 1130
55.10.2 Deleting a Range Line 1132
55.10.3 Configuring Range Lines 1132

55.1. Intervisibility (Line-of-Sight)

You can display intervisibility (line-of-sight or LOS) between points, within areas, and between a point and all simulation objects in a defined area. Intervisibility displays can be transient or permanent. Permanent intervisibility displays are objects. You can resize them and move them around the terrain. Intervisibility objects can be tied to a particular simulation object.

You can display a terrain profile for point-to-point intervisibility lines.

Figure 429. Intervisibility



You can configure parameters for intervisibility objects, such as the height above the terrain from which intervisibility is calculated. You can also change the colors used to display intervisibility.

- By default, intervisibility between simulation objects is calculated from the center of their bounding volumes. It is possible that this may not match the line-of-sight calculations used by the simulation application whose output you are viewing. If so, simulation objects may shoot at targets for which they do not appear to have line-of-sight. The VR-Forces target detection sensor uses an offset, which is often above the center of the entity. Unless you change the Origin LOS Elevation value in the Intervisibility Options dialog box to use the same offset as the target detection sensor, simulation objects may shoot at targets for which they do not appear to have line-of-sight.
- Linear and areal features do not block line-of-sight. Only point features block LOS.

55.2. Types of Intervisibility

VR-Forces supports these types of intervisibility:

- Point-to-point (intervisibility line).
- Intervisibility fan.
- Entity intervisibility.

The intervisibility fan and entity intervisibility are displayed as a circular area. However, intervisibility calculations are made within a cylindrical area extending above and below the terrain, so that intervisibility calculations can be made for air platform entities and subsurface entities.

In the 2D view, intervisibility lines are fully visible on the terrain. In the 3D view, they may pass through the terrain.

55.2.1 Point-to-Point Intervisibility

Point-to-point intervisibility (Figure 430) shows the line-of-sight between any two points. The portion of the line that is visible from the starting point is green. The portion that is not visible is red. You can change the colors used.

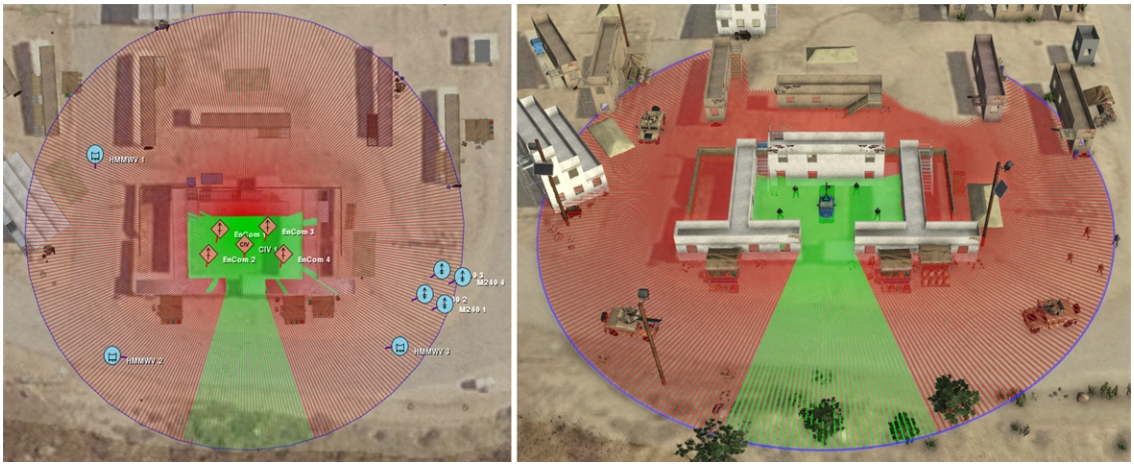
Figure 430. Point-to-point intervisibility - 2D and 3D



55.2.2 Intervisibility Fan

An intervisibility fan (Figure 431) shows a series of lines radiating from the center of a circle to its outer edge. Each ray shows intervisibility along its path from the origin to the border. You can configure the distance between rays. (For details, see "55.5. Configuring Intervisibility Lines and Fans" on page 1127.)

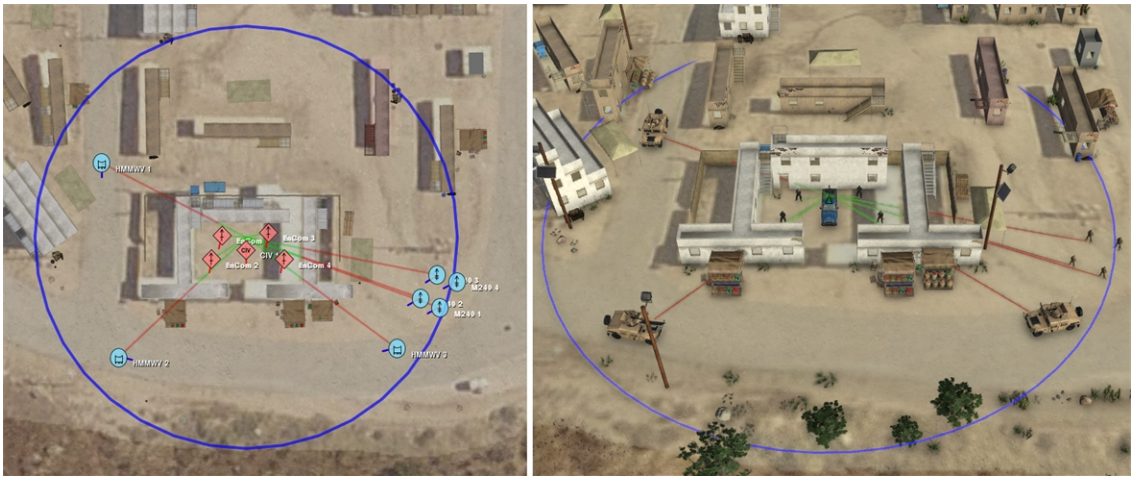
Figure 431. Intervisibility fan - 2D and 3D



55.2.3 Simulation Object Intervisibility

Simulation object intervisibility (Figure 432) shows which simulation objects are visible from the center of a defined area and whether the simulation objects can see the center point. If you center the intervisibility circle on a point on the terrain, you can see which simulation objects can see that point. If you lock the circle to a simulation object, you can see which simulation objects can see that object and which it can see. (For details, see "55.5. Configuring Intervisibility Lines and Fans" on page 1127.)

Figure 432. Simulation object intervisibility - 2D and 3D



55.3. Intervisibility Objects Can Be Transient or Permanent

You can configure intervisibility objects to be transient or permanent. (For details, see "55.5. Configuring Intervisibility Lines and Fans" on page 1127.) A transient intervisibility object displays intervisibility from the origin point (defined by a mouse click) to wherever you move the cursor. If you click the mouse again, you reset the origin. You cannot use the

mouse for other actions until you exit intervisibility mode (by right-clicking). This mode is convenient when you want to quickly check intervisibility, but do not want to keep intervisibility objects in the display.

A permanent intervisibility object stays in the display after you create it. You can resize the object, edit drawing options, and move it around the display. Intervisibility objects are not saved as part of the scenario.

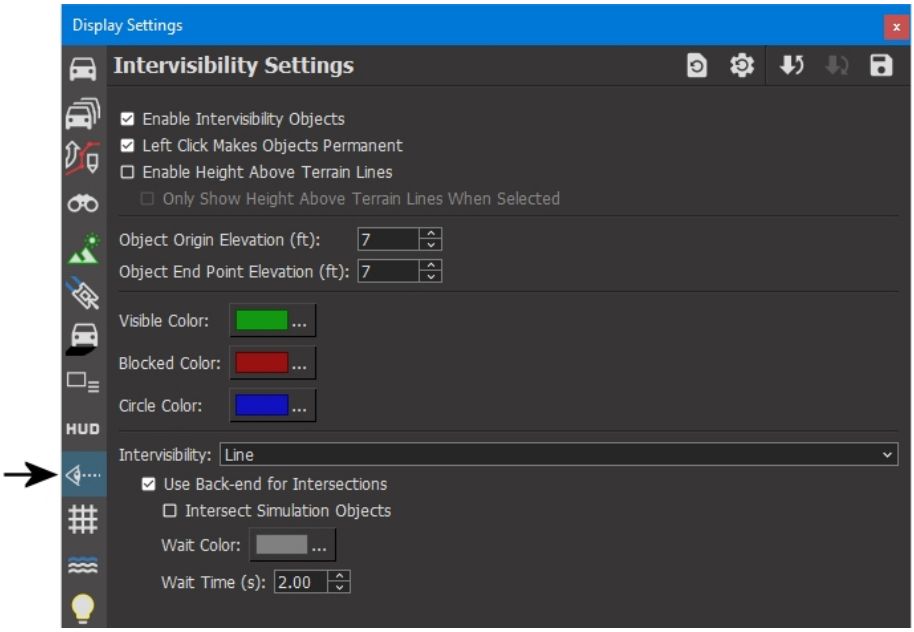
55.3.1 Creating a Transient Intervisibility Object

Transient intervisibility objects do not stay in the display after you quit intervisibility mode.

➤ **To display transient intervisibility:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 433).

Figure 433. Intervisibility Settings page



3. Clear the Left Click Makes Objects Permanent check box.
4. Choose **Intervisibility > Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar.
5. Click on the terrain to place the origin of the intervisibility object.
6. Move the cursor around the terrain to expand and contract the object. For point-to-point intervisibility, moving the cursor also changes the direction of the line.
7. To change the origin of the intervisibility display, click another location on the terrain.
8. Right-click to cancel intervisibility mode.

55.3.2 Creating a Permanent Intervisibility Object

Permanent intervisibility objects remain in the display after you create them. (However, they are not saved as part of the scenario.)

➤ **To create a permanent intervisibility object:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 433 on the previous page).
3. Select Enable Intervisibility Objects.
4. Select Left Click Makes Objects Permanent.
5. Choose **Intervisibility > Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar.
6. Click on a simulation object or on a location on the terrain where you want the origin of the intervisibility object to be located. An intervisibility object displays. As you move the cursor closer to the origin, or farther away from it, the circle or line contracts and expands. If you are displaying point-to-point, you can rotate the line around the origin before you specify the end point.
7. Click on the terrain where you want to fix the end point of a line or border of a circle.

55.4. Editing an Intervisibility Object

You can change these aspects of an intervisibility object:

- The location of either end point of an intervisibility line, to change its length and direction.
- The origin of an intervisibility fan, thereby moving the entire fan.
- The diameter of an intervisibility fan.

ⓘ If an intervisibility object is anchored to a simulation object, you cannot change the origin.

➤ **To change the length or diameter of an intervisibility object:**

1. To edit a line, double-click its end point.

   To edit a fan, select the object, then double-click the outer ring of the fan.

   The cursor changes to ✣ (a four-way arrow).
2. Drag the end point or circle towards or away from the origin. If you are editing an intervisibility line, you can move the end point to any location.
3. Click to exit edit mode.

ⓘ To revert to the original size of the intervisibility object, right-click while you are still in edit mode.

➤ **To move an intervisibility fan or line:**

1. Double-click a fan's center point or any place on a line. The cursor changes.
2. Drag the object to a new location.
3. Click to set the location.

To revert to the original location of the intervisibility object, right-click while you are still in edit mode.

55.5. Configuring Intervisibility Lines and Fans

You can configure attributes of intervisibility objects:

- The elevation of the origin and end point.
- The sampling rate for intervisibility fans.
- The update rate for entity intervisibility fans.
- Display of height above terrain lines (3D) and height labels (3D and 2D).
- The colors used to display intervisibility objects.

When you change colors or the sampling rate, changes affect all existing intervisibility objects. Changes to the origin and end point elevation only affect newly created objects.

VR-Forces can calculate intersections in the GUI or engine.

➤ **To configure intervisibility objects:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 433 on page 1125).
3. For Intersection configuration, select the intervisibility object to configure in the Intervisibility list.
4. Change the settings as desired. Table 46 describes the settings.

Table 46. Intervisibility settings

Option	Description
Enable Intervisibility Objects	Enables the creation of intervisibility objects.
Left Click Makes Objects Permanent	Enables permanent intervisibility objects. See "55.3.1 Creating a Transient Intervisibility Object" on page 1125 and "55.3.2 Creating a Permanent Intervisibility Object" on the previous page.
Enable Height Above Terrain Lines	Enables or disables display of height above terrain lines (3D) and height labels (3D and 2D).

Table 46. Intervisibility settings (continued)

Option	Description
Only Show Height Above Terrain Lines Only for Selected	If selected, HAT lines are displayed only for the selected intervisibility objects.
Object Origin Elevation	The height, in meters, above the terrain at the origin point from which to calculate intervisibility.
Object End Point Elevation	The height, in meters, above the terrain at the end point to use to calculate intervisibility.
Visible Color Blocked Color	The colors to use for the portion of an intervisibility line that a simulation object can see and the portion where visibility is blocked.
Circle Color	The color to use for the outer circle of an intervisibility fan.
Intersections	
Intervisibility	You can select Line, Fan, or Object Fan.
Use Sim Engine for Intersections	Select this check box to calculate intersections based on the sim engine. Clear this check box to use the GUI.
Intersect Simulation Objects	Select this check box to include simulation objects when calculating intervisibility in the sim engine.
Wait Color	Specify the color to display on simulation objects while VR-Forces calculates intervisibility.
Wait Time	Specify the delay after an entity stops moving to query the sim engine for intersection information. If the entity begins moving again before the information is received from the sim engine, VR-Forces ignores the sim engine result. If you set the time to 0, you could flood the network with requests and impact performance.
Fan Line Sample Interval	The distance, in degrees, between each ray in a intervisibility fan. After changing this value, press **Enter** or click away from the field. (This field is only available when you select Fan from the Intervisibility list.)

55.6. Pinning an Intervisibility Object to a Simulation Object

When you pin an intervisibility object to a simulation object, the intervisibility object moves with the simulation object and you know that its origin is centered on the simulation object.

➤ **To pin an intervisibility object to a simulation object:**

1. Select the simulation object to which you want to pin the intervisibility object.
2. Choose **Intervisibility > Create** *intervisibility_type*, where *intervisibility_type* is a type of intervisibility object, or click one of the buttons on the Intervisibility Toolbar. The origin is placed on the simulation object icon or model.
3. Specify the end point for the intervisibility object.

55.7. Showing and Hiding Intervisibility Lines

You can toggle the display of intervisibility lines on and off.

➤ **To enable or disable the display of intervisibility lines:**

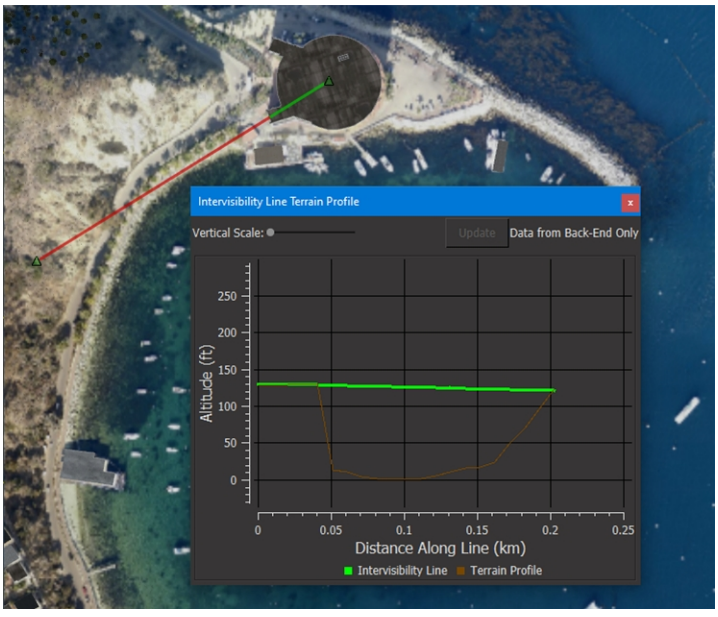
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Intervisibility Settings page (Figure 433 on page 1125).
3. Select or clear the Enable Intervisibility Objects check box.

> You can also enable and disable display of intervisibility objects by choosing **Intervisibility > Enable/Disable Intervisibility**, or by clicking the Enable/Disable Intervisibility button ( ) on the Intervisibility Toolbar.

55.8. Displaying an Intervisibility Line's Terrain Profile

By default, a terrain profile window displays automatically when you create an intervisibility line. The terrain profile can be very helpful when you try to understand how terrain changes are affecting intervisibility. Figure 434 shows an obstructed intervisibility line and its terrain profile, whose graph illustrates the line intersecting the terrain.

Figure 434. Terrain profile for an intervisibility line



You can disable this setting on the Display Settings dialog box, Terrain Profile Settings page. For details, see"65.1.2 Configuring Terrain Profiles" on page 1267.

55.9. Deleting Intervisibility Objects

Before you delete an intervisibility object, note that:

- If you delete a simulation object that has an intervisibility object pinned to it, the intervisibility object is also deleted.
- If an intervisibility object is pinned to a simulation object, you can delete the intervisibility object without affecting the simulation object by selecting the outer circle or end point and then deleting it.

➤ To delete an intervisibility object, select the object and choose **Intervisibility > Delete** (or press **Delete**).

55.10. Range Lines

During a session, range lines allow you to see the great circle distance and bearing between a start and end point in the Plan View (2D). The distance between the two points and the bearing from the start point to the end point display at the midpoint of the line. If the end points move, for example, the simulation object it is attached to is moving, then this information changes to reflect the change in distance and heading.

See "55.10.1 Creating a Range Line" below and "55.10.3 Configuring Range Lines" on page 1132 for details about using this feature.

55.10.1 Creating a Range Line

You can create a range line in the Plan View to see the great circle distance between two points and also the bearing from the start point to the end point.

When you create a line from the context menu or Create menu, you can place each end of the line on a point in the terrain or attach it to a simulation object. If either end of a range line is attached to a simulation object, the distance and bearing information updates as the simulation object moves.

You can also quickly draw a range line using the middle mouse button to measure a distance without connecting it to any objects. This is useful when you want to quickly check the distance and bearing between two points.

Figure 435. Placing a range line on the terrain



Figure 436. Attaching a range line to a simulation object



➤ **To draw a quick range line:**

1. Position the cursor at the position you want to measure from and then press and hold the middle mouse button.
2. Move the cursor to the position you want to measure to. Observe the information displayed beside the line.
3. When you are finished, release the middle mouse button. The line disappears.

➤ **To create a range line from the context menu:**

1. Right-click on a point in the terrain or on a simulation object. If you click on a simulation object, that is the start point for the range line.
2. From the context menu, choose **Create > Create Range Line**. If you clicked on a simulation object, continue to step 4.
3. Click to place the start point on a place in the terrain or on a simulation object (if one is not already selected).
4. Click to place the end point. You can also place it on the terrain or a simulation object.

➤ **To create a range line from the Create menu:**

1. Choose **Create > Create Range Line**. If a simulation object is selected, that is the start point for the range line. Continue to step 3.
2. If you did not have a simulation object selected, position the cursor where you want the range line to begin. If you want to attach it to a simulation object, place the cursor over that object until its outline turns white and then click. To place the start point on the terrain, simply click on the location.
3. Position the cursor where you want the range line to end and then click. You can attach the end point to a simulation object or a point on the terrain. The range line persists until you delete it or you rewind the scenario.

55.10.2 Deleting a Range Line

➤ **To delete a range line:**

1. Select the line.
2. Choose **Intervisibility > Delete**.

ⓘ You can also use the context menu to choose **Intervisibility > Delete**.

55.10.3 Configuring Range Lines

You can configure the attributes of range lines using the Model Definition Editor. For details about editing model definitions, see "Creating and Editing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.)

- The font and offset for the measurement text.
- The colors used to display range lines and text.
- The thickness and style of the range lines.
- The font size of the distance and bearing data.
- Whether the measurements round up to a higher unit (for example, meters to kilometers).

➤ **To configure range lines:**

1. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
2. Select the Model Definition Editor page.
3. Select OverlayRangeLine.
4. Change the settings as desired.

- Range lines and bearing use the distance measurement for the application, as specified in Display Units Settings. See "84.9. Specifying Display Units" on page 1634.
- If you enable AreSmartDistancesForMeasurementTextEnabled, VR-Forces rounds the measurement up. For example, if the range measurement is set to meters and the line is longer than a kilometer, the display unit changes to kilometers.



56. Configuring Emitter Volumes

This chapter explains how to configure display of emitter volumes. For information about how to configure simulation of emitters, see "Chapter 80. Configuring Emitters and Radios" on page 1571.

56.1. Emitter Volumes Overview 1136
56.1.1 Configuring Emitter Volume Color 1136
56.1.2 Controlling Emitter Volume Radius 1137
56.1.3 Configuring Emitter Volume Segments 1137

56.1. Emitter Volumes Overview

You can configure how VR-Forces colors emitter volumes, how they are drawn, and how they are scaled for visibility. You can configure emitter volumes by editing the DefaultEllipsoidArc model definition or by creating customized ellipsoid arc model definitions and applying them to different entity models.

For an entity to display emitter volumes in 2D observer modes and 3D observer modes, it must have a 2D Emitter Volume visualizer in its 2D model set and 3D Emitter Volume visualizer in its 3D model sets. The DefaultEllipsoidArc model definition is an attribute of the visualizer.

56.1.1 Configuring Emitter Volume Color

By default, emitter volumes are displayed using fixed colors for high frequency and low frequency. You can configure emitter volumes to be colored based on the emission frequency or pulse rate. When emitter volume color represents frequency, color is determined by the frequency field of the EM Emission update message. In both cases, you specify a low frequency or pulse rate value and a high value.

If the emitter frequency or pulse rate is lower than your specified low value, the emitter volume is red. As frequency rises above the specified low value, emitter volume color changes, progressing through the colors of the visible spectrum until the specified high value is reached. At the high value and above, the emitter volume is violet.

To change coloring from the default, you must add the appropriate attributes to the emitter volume visualizer for the entities whose model definitions you want to change. Table 47 describes the emitter volume visualizer attributes that affect emitter volume color.

Table 47. Emitter volume visualizer attributes

Attribute	Description
colorByFrequency	If True, color by frequency instead of using static colors.
lowFrequency	The low frequency threshold for coloring by frequency. Default: 800 MHz.
highFrequency	The high frequency threshold for coloring by frequency. Default: 1 GHz.
usePulseFrequency	If **colorByFrequency** is True and **usePulseFrequency** is True, color emitter volumes by pulse frequency.
lowPulseFrequency	The low pulse rate threshold for coloring by frequency. Default: 100 Hz.
highPulseFrequency	The high pulse rate threshold for coloring by frequency. Default: 3 KHz.

56.1.2 Controlling Emitter Volume Radius

An emitter volume's radius (that is, the distance it extends from the emitter) is based in part on emitter power and in part on elevation and azimuth sweep data obtained from EM-Emission state messages. A higher emitter power value results in a longer radius, while a wider field of view results in a shorter radius (because the radiated power is distributed over a wider area).

If VR-Forces relied exclusively on these values, the range of sizes among different emitter types could become too wide to be practical, with some emissions being barely visible, and other emissions covering the entire virtual world. Therefore, a configurable exponent weight and scale factor have been added to help fine tune the emitter beam radii. The radius exponent weight (a value between 0 and 1) slows the growth of the radius when power or area is greater than 1 and increases the growth of the radius when power or area is less than 0. The scale factor increases the radii of all beams by the given value.

To change the scale factor and exponent from the default, you must add one or both of these attributes to the emitter volume visualizer for the entities whose model definitions you want to change:

- linearScale. The amount of linear scaling applied to all emitter volumes. Default: 100.
- exponentialScale. The amount of exponential scaling applied to emitter volumes. Default: 0.1.

How VR-Forces Calculates the Radius

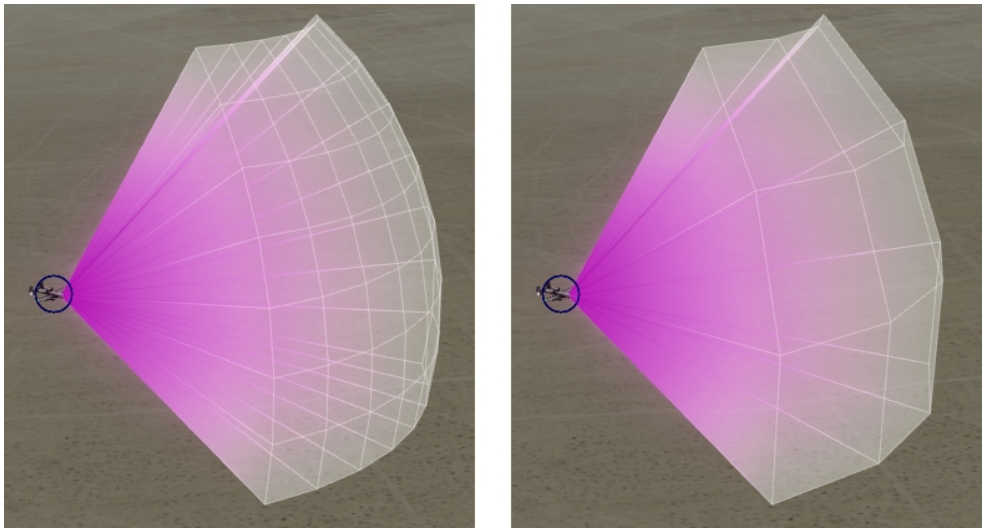
VR-Forces calculates emitter volume radius by approximating the spread of the emissions based on the elevation and azimuth sweeps. If the approximate coverage is zero, it sets the value to 1. Then it applies the linear and exponential scaling values. The code for implementing this is:

```
double tempVar = 8*elevationSweep*sin(2*azimuthSweep);
if (tempVar <=0) tempVar=1.0;
scale = linearScale * pow(sqrt(power/tempVar), exponentialScale );
```

56.1.3 Configuring Emitter Volume Segments

VR-Forces displays an emitter volume as a group of spherical sections projecting from the sensing device. By default, for each 10 degrees (or portion thereof) in the emitter's field of view, VR-Forces draws one segment. Figure 437 shows segments set to 10º (the default) and 30º.

Figure 437. Emitter volume segments - 10° and 30°



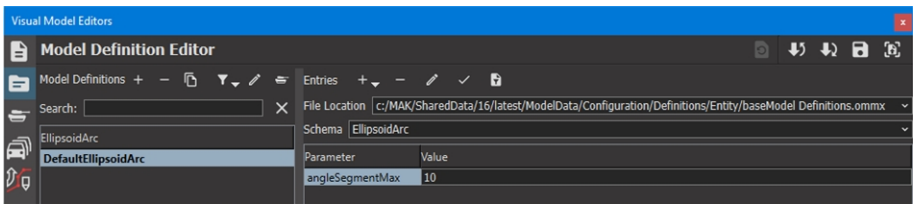
Emitter volume segments are managed by the EllipsoidArc schema and model definitions based on it. VR-Forces ships with one ellipsoid arc model definition – DefaultEllipsoidArc. You can edit this model definition or create copies of it and customize them. If you want to have different emitter volumes for different entities, assign a different ellipsoid arc model definition to each one.

Specifying the Segment Angle

➤ **To specify the segment angle:**

1. Decide whether you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, see "Model Definition Files" in the *Adding Content to MAK ONE Applications Reference Manual*.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 438).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, see "Importing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.)
5. Click the Filter button ( ) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc (Figure 438).

Figure 438. EllipsoidArc model definition

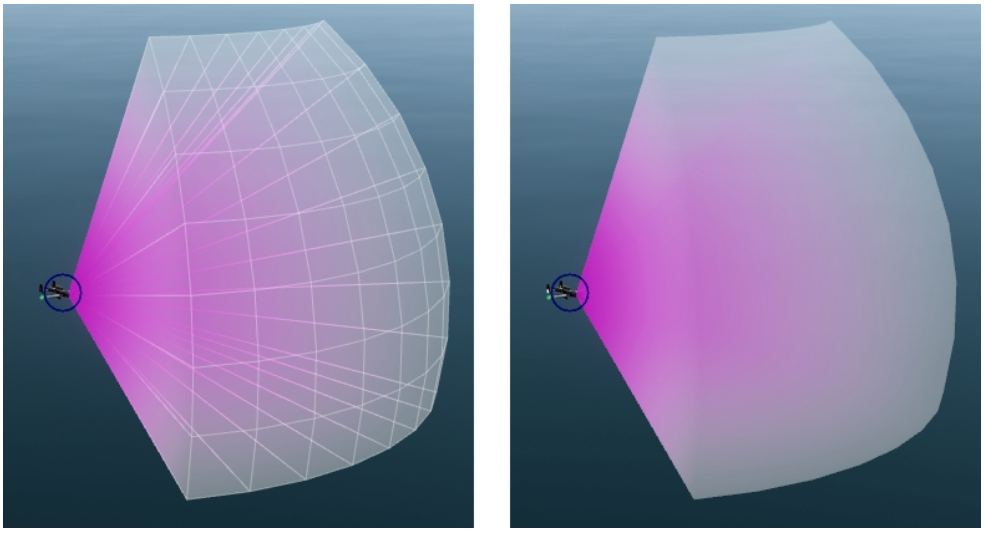


7. Click the Add a new parameter button (+▾) and select **angleSegmentMax**.
8. Change the value of the **angleSegmentMax** parameter. The minimum value is 10 (degrees); the maximum is 45.
9. Export the updated setting to the loaded definition file or to a separate file.

Displaying Segment Outlines

By default, VR-Forces displays opaque outlines showing the segment edges along the surface of the emitter volume.

Figure 439. Segment outlines - on and off



➤ **To display or hide segment outlines:**

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, see "Model Definition Files" in the *Adding Content to MAK ONE Applications Reference Manual*.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page.
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, see "Importing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.)
5. Click the Filter button (▼▾) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc.
7. Change the value of the **showLines** parameter.
8. Export the updated setting to the loaded definition file or to an individual file.

Adding a Custom EllipsoidArc Model Definition

If you want to customize the segment angle and segment outlines for a particular entity without changing the values for other entities, you must create a separate ellipsoid arc model definition and add it to the entity's model definition.

➤ **To create a custom ellipsoid arc model definition:**

1. Decide if you want to export the edited model definition to its own file or to an existing file. (For details about the model definition work flow, see "Model Definition Files" in the *Adding Content to MAK ONE Applications Reference Manual*.)
2. Choose **Settings > Visual Model Editors**. The Visual Model Editors dialog box opens.
3. Select the Model Definition Editor page (Figure 438 on page 1138).
4. If you want to export the edited model definition to an existing file, use the Load and Replace procedure to load the file. (For details, please see "Importing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.)
5. Click the Filter button ( ) and select EllipsoidArc. The ellipsoid arc model definitions are listed in the window.
6. Select DefaultEllipsoidArc.
7. Click the Duplicate Model Definition button ( ). The Duplicate Model Definition dialog box opens with a default definition name.
8. Accept the generated name or type a new name for the model definition.
9. Click OK. The model definition is added to the list.
10. Change the parameter values for the new model definition as desired.
11. Select the Entity Definition Editor page.
12. Select the entity whose emitter volume you want to change.
13. Select the 3D Sensor Volume visualizer for the entity.
14. Click the Value column for the **modeldefinition** attribute. The Choose Model Definition dialog box opens. It has a list of model definitions.
15. Select the model definition that you just created.
16. Click OK.
17. Export the updated setting to the loaded definition file or to an individual definition file.
18. If you are connected to an exercise, disconnect and then reconnect to enable the new model definition.



57. Sound Effects

VR-Forces can play sound effects for entities, fire events, and detonation events.

57.1. Introduction to Sound Effects 1142
57.2. Enabling Sound Effects 1142
57.2.1 Configuring Sound Effects 1143
57.2.2 Decibels Configuration 1144
57.3. Associating Sound Files with Entities and Events 1145

57.1. Introduction to Sound Effects

VR-Forces can associate sound effects with entities, fire events, and detonation events. You can enable and disable use of sound effects and you can specify settings that determine how sounds are perceived.

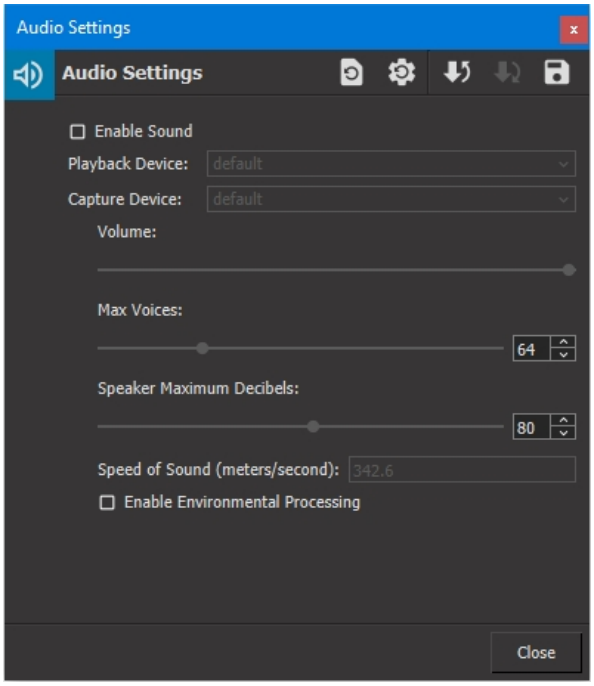
Using sound effects can degrade performance.

57.2. Enabling Sound Effects

➤ **To enable sound effects:**

1. Choose **Settings > Audio Settings**. The Audio Settings dialog box opens (Figure 440).

Figure 440. Audio Settings page



2. Select the Enable Sound check box.
3. Optionally, specify the playback device (speaker) and capture device (microphone).

You can also enable and disable sound effects by clicking the Audio button ( ) on the Display Settings toolbar.

57.2.1 Configuring Sound Effects

You can configure the sound settings.

- Playback Device. Select the speaker for the audio output.
- Capture Device. Select the microphone to use. This is typically for use with MAK Radio.
- Volume. This is the volume of the system.
- Max Voices. You can have from 1 to 256 voices, or maximum simultaneous playing sounds. Sounds that play at the same time are mixed into a single sound wave prior to being sent to the speaker. This mixing has a CPU cost. A high number of allowable voices can affect performance when there are many audio sounds trying to play.
- Speaker Maximum Decibels. Calibrates the audio system for the speakers that the system is played on. This affects how sounds at a distance fall off so that the sound loudness is realistically replicated. The default setting of 80 db represents headphones or a desktop PC's speakers. See "57.2.2 Decibels Configuration" on the next page for more information.
- Speed of Sound. This is the speed of sound in the atmosphere in meters/second. The default value is derived by using a standard day as parameters (20° C, 50% relative humidity, 1013.25 mbar/29.92 inHg). This affects the delay in playing sounds at a distance. For example, an explosion 2 km away has a several second delay from the time you can see the visual effect to when you can hear the audio effect.
- Enable Environmental Processing. Enables low pass filtering. As sound propagates through air, higher frequency sounds are more absorbed the further you move from the source of the sound. This setting replicates air absorption of a sound based on the distance the sound is from the listener.

➤ **To configure sound settings:**

1. Choose **Settings > Audio Settings**. The Audio Settings dialog box opens (Figure 440 on the previous page).
2. To select the speaker, choose a device from the Playback Device list.
3. To select the microphone, choose a device from the Capture Device list.
4. To set the volume, adjust the Volume slider.
5. To specify the maximum number of voices, adjust the Max Voices slider or change the value in the text box.
6. To set the speaker maximum decibels, adjust the Speaker Maximum Decibels slider or change the value in the text box.
7. To set the speed of sound, type a value in the Speed of Sound box.
8. To enable lowpass filtering, select the Enable Environmental Processing check box.

57.2.2 Decibels Configuration

As a feature, the VR-Forces audio system uses a sound decibel scaling system to compensate for the deployed sound systems such as headphones, computer speakers, and amplifiers. This helps provide a similar audio experience regardless of the system that you use.

You can configure each individual entity sound to match real world characteristics. For example:

- A whisper or rustling of leaves registers at 20 dB.
- A conversation at home is approximately 50 dB.
- A passenger car traveling on the freeway at 50 feet away is 70 dB.
- Inside a helicopter cabin 100 dB.

When configuring the decibels (dB) for a sound, you should select a decibel level that represents the level near the sound source. The dB setting should be the loudest level that you would ever want to hear the sound, which might differ from the real-life level. This allows the audio system to calibrate the sound levels automatically.

For each sound, based on the Speaker Maximum Decibels setting, the audio system computes a sound "reference" distance and "falloff" distance:

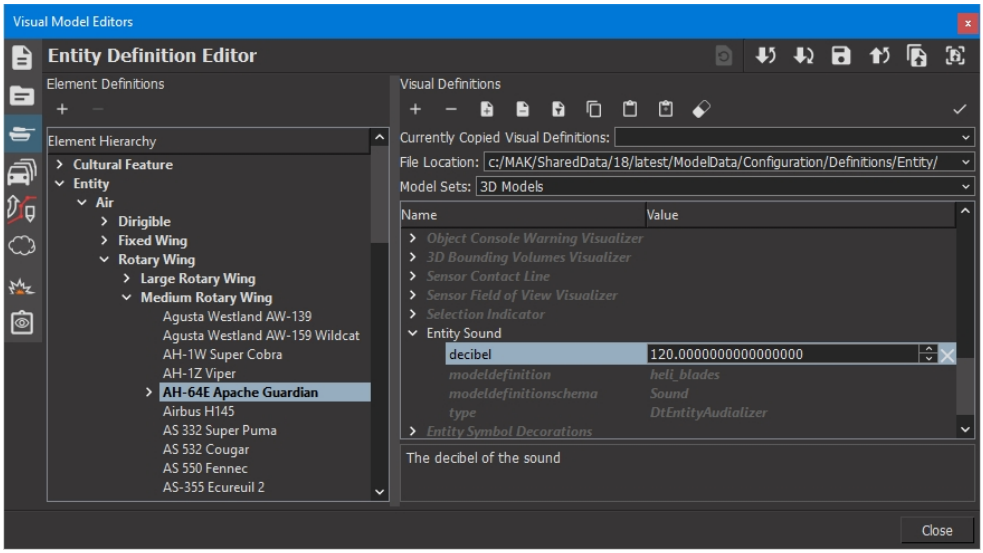
- The reference distance is the distance at which the sound is the loudest it can be heard.
- The falloff distance is the distance at which the sound can no longer be heard.

These examples show when a VR-Forces entity's sound is configured to be 80 dB and the observer is close to the entity:

- If the Speaker Maximum Decibels is set to 80 dB or lower, then the sound will be played and heard at its maximum volume.
- If the Speaker Maximum Decibels is set to 120 dB, then the sound will be played at a lower volume because the speaker system is expected to increase the heard volume to 80 dB.

Other sound effects settings are covered in "57.2.1 Configuring Sound Effects" on the previous page.

Figure 441. Example of entity decibel attribute in Entity Definition Editor



57.3. Associating Sound Files with Entities and Events

VR-Forces includes a set of sound files (WAV format) for generic entity platforms and for fire and detonation events. The files are in *./SharedData/19/latest/Audio/Sounds*. You add and edit mappings between entity types, fire events, or detonation events and sound files in the Visual Models Editor. Sound files are associated with model definitions. For example, the **filename** parameter for the heli_blades model definition specifies *$(SHARED_DATA_DIR)/Audio/Sounds/heli_blades.wav*. The Rotary Wing element definition has a parameter called **Entity Sound**. The value for the **model definition** attribute is **heli_blades**.

You can quickly see which model definitions are available for sounds by filtering the list of the Visual Model Editors, Model Definitions Editor page for the Sound schema.



58. MAK Radio

This chapter describes the MAK Radio features. There is a dialog box, which can also be run as a standalone application, and a toolbar.

58.1. MAK Radio Overview 1148
58.2. Using the Radio in VR-Forces 1148
58.2.1 MAK Radio Command-Line Options 1149
58.2.2 Operating the Radio from the Dialog Box or Application 1151
58.2.3 Operating the Radio from the Radio Control Toolbar 1153
58.3. Using the Intercom or a VOX Radio 1154
58.4. Configuring the MAK Radio 1154
58.5. Audio Settings 1158
58.6. Jamming Radios 1159

58.1. MAK Radio Overview

VR-Forces supports voice communications over radios and intercoms. You can communicate with anyone using the radio application that is in the same exercise. There are 2 radios, each of which has 2 channels. (The radios are configurable. For details, see "58.4. Configuring the MAK Radio" on page 1154.) Two applications can communicate only when the radios are tuned to the same frequency and that frequency is not being jammed.

There is also an intercom, which represents the internal intercom used by crew members of a single vehicle to communicate with each other. For two applications to communicate using the intercom, they must both be part of the same exercise and have the Attached Entity field set to the same entity. The ADMIN option indicates that the intercom is not connected to any entity but instead to all MAK Radio users who also specify ADMIN. For information about using the intercom and voice-activated radio, see "58.3. Using the Intercom or a VOX Radio" on page 1154.

Audio data from the radio system is sent as signal interactions/PDUs on the simulation network. It can be received in other VR-Forces GUI applications, the standalone applications, or other applications that can receive standard signal interactions/PDUs (such as VR-Forces).

Radio communications are perfect. Radio signals are not affected by terrain, weather, distance, or other factors. However, you can use the standalone radio application or the radio panel in VR-Forces to jam radios. For more information about jamming, see "58.6. Jamming Radios" on page 1159.

The radio uses the audio devices configured in the Audio Settings dialog box for capture and playback. See "57.2.1 Configuring Sound Effects" on page 1143 for details.

58.2. Using the Radio in VR-Forces

Each configured radio has a radio head that represents its state. It displays the radio name, its frequency, and the audio levels for transmit (TX) and receive (RX).

There are different ways to use the radio: the dialog box or toolbar in VR-Forces, or the standalone application.

➤ To open the MAK Radio dialog box, choose **View > Radios**. The Radio Communications dialog box opens (Figure 442 on page 1153).

➤ To open the MAK Radio toolbar, choose **View > Toolbars > Radio Control Toolbar**. The toolbar displays (Figure 443 on page 1153).

➤ **To start the MAK Radio standalone application from the Windows Start menu:**

1. Choose **MAK Technologies > VR-Forces 5.2 Tools**.
2. In the Tools folder, double-click the MAK Radio shortcut. The Simulation Connection Configurations dialog box opens.
3. Specify the connection information of the simulation where you want to use the radio.
4. Click Launch. The MAK Radio application opens (Figure 442 on page 1153).

➤ **To start the MAK Radio standalone application from the command line:**

1. Change the directory to *./bin64* and enter `--makRadioApp`*protocol* where *protocol* is `DIS`, `HLA1516e`, or `HLA4`. You must also specify the connection information of the simulation where you want to use the radio. See Table 48 below for the list of command-line options.
2. Run the command. The MAK Radio application opens (Figure 442 on page 1153).

58.2.1 MAK Radio Command-Line Options

Table 48 lists the MAK Radio standalone application's command-line options.

Table 48. makRadioApp command-line options

Option	Description
(`--appNumber` | `-a`) *number*	Specifies the application number
(`--destAddrString` | `-A`) *address*	the destination address
`--dis5Timestamps`	In DIS 7, there is a discrepancy in timestamps that can lead to some of them appearing to take place in the future. Use this option to enable the time management method from DIS 5.
`--fileNotifyLevel` *level*	Notify level in to use in the log file.
`--lightMode`	The MAK Radio application loads in light mode rather than the default of dark mode.
`--logFileName`	Specifies the log file name.
`--mcastTtl` *time*	Specifies the multicast time-to-live. Default: -1.
`--mimModules` *HLAE_MIM*	Specifies an HLA Evolved MIM module.
`--netnFomRevision` *revision_number*	Specifies the internal revision of the implementation of the NETN FOM version as an integer. The default (and currently the only supported revision) is 1.
`--netnFomVersion` *version_number*	Specifies the NETN FOM version as a double. The default (and currently the only supported version) is 3.0.
`--noRtiCompilerCheck`	Disables the check of the RTI compiler version.
(`--notifyLevel` | `-n`) *notification_level*	Specifies the notification level for application messages, where *notification_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, see "5.4.3 Setting the Notification Level" on page 189.
`--outputInterval` *interval*	Interval for output.
`--radioConfig` *filename*	Specifies the radio configuration file to use.

Table 48. makRadioApp command-line options (continued)

Option	Description
`--recvBufferSize` *size*	Receive buffer size.
`--rprFomRevision` *revision_number*	RPR FOM revision for RPR FOM 2.0, draft 17.
`--rprFomVersion` *version_number*	RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.0006, 2.0014, 2.0017, or 2.0)
`--sendBufferSize` *size*	Send buffer size. Default: -1.
(`--siteId` | `-s`) *ID*	Specifies the site ID.
`--suppressSelfReflect`	Represses self reflection of the DIS exercise connection.
`--useAsyncIO`	Use asynchronous I/O. Default: False.
`--useIpv6`	Specifies that the DIS exercise connection use the IPV6 protocol, if available.
`--useUserSettingsDirectory`	Indicates that the system should store settings in the user login settings directory, rather than the application settings directory. Use in conjunction with `--sharedSettingsDirectory`.
HLA Only	**These options are available when you run makRadioAppHLA1516e or makRadioAppHLA4.**
`--exConnConfigFile` *filename*	Specifies the exercise connection configuration file to use. Use this to override settings in the VR-Link exercise connection configuration file, *./appData/settings/connections/MAK-ONE-YYYY-Config.xml*, which defines the default connections settings. For more information, refer to the *MAK ONE Interoperability Guide*.
(`--execName` | `-x`) *exec_name*	Specifies the name of the federation execution. Default: MAK-RPR-2.0.
(`--federateName` | `-N`) *federate_name*	Specifies a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI automatically assigns a unique name for the federate. Default: VR-Forces. (HLA Evolved only)
(`--federateType` | `-p`) *type*	Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.
(`--fedFileName` | `-F`) *fedfile_name*	FED file name.
`--fomMapperInitData` *data*	FOM Mapper initialization data.

Table 48. makRadioApp command-line options (continued)

Option	Description
(--fomMapperLib | -f) *libname*	FOM Mapper library name.
--fomModules *module*	Specifies an HLA Evolved or HLA 4 FOM module. Can be used multiple times.
--ignoreAdvisories	Enables or disables HLA advisory messages. Default: true.
(--localSettingsDesignator | -S) *designator*	Specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage.
--setFomModuleList	Overwrite FOM Modules with the listed modules. Accepts a comma-separated list of module names; the best practice is to list them in hierarchical order to prevent any issues related to dependencies.
(--ignore_rest | --)	Ignores all command-line options that are listed after this argument. This option is useful for disabling part of a command line in a script for test purposes so that you do not have to retype all of the command options at a later point.
(--version | -v)	Displays version information.
(--help | -h)	(Displays command-line usage information in a console window.)
DIS Only	**These options are available when you run makRadioAppDIS.**
(--disPort | -P) *port*	DIS port.
--disVersion *version*	The DIS version. Default: 7.
(--exerciseId | -x) *ID*	Exercise ID.
(--multicastAddresses | -S) *address1 ... addressn*	Specifies one or more multicast addresses.

58.2.2 Operating the Radio from the Dialog Box or Application

The controls for operating the radio are the same in the dialog box within VR-Forces and the standalone application (Figure 442).

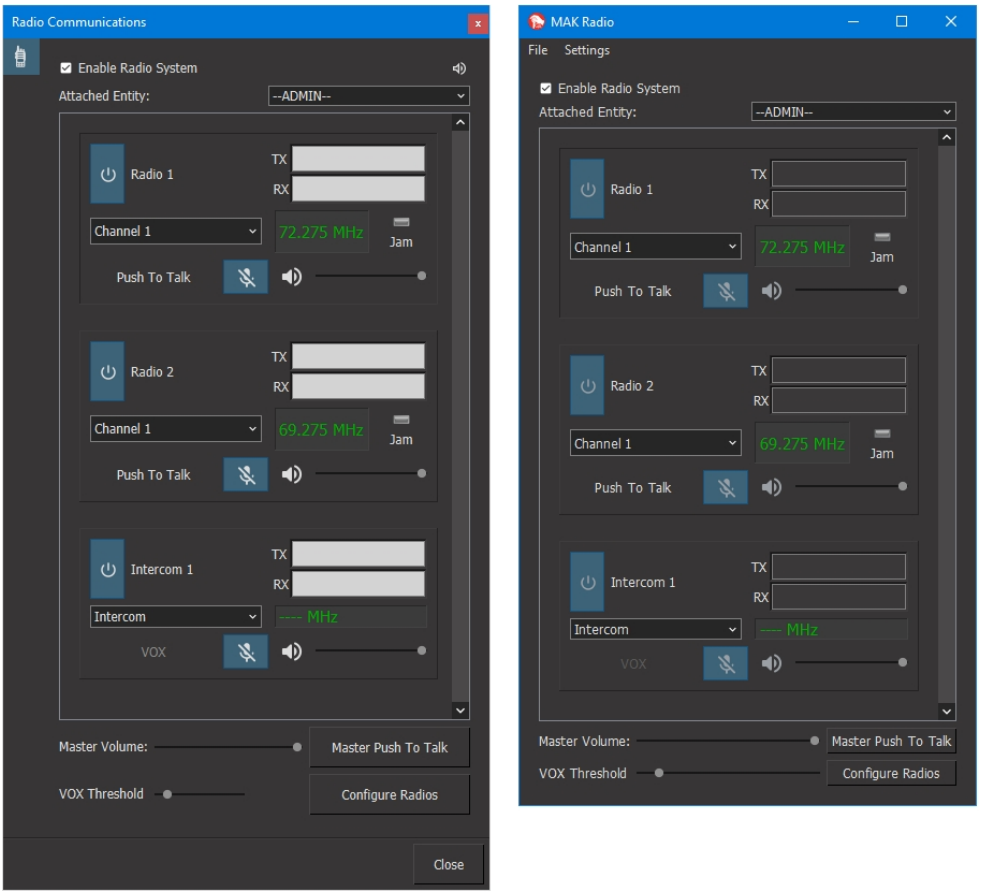
The default key for Push to Talk is **T**, but in the standalone application you can specify other keys using the Audio Settings. See "58.5. Audio Settings" on page 1158.

➤ **To use the radio:**

1. In the Radio Communications dialog box or MAK Radio standalone application, select the Enable Radio System check box.
2. In the Attached Entity list, select the entity you want to speak for, or ADMIN if you are an observer and want to talk to everyone.
3. Optionally, adjust the Master Volume slider to adjust the volume of all radios.
4. Optionally, adjust the VOX Threshold slider to adjust the level of sound needed to enable voice activated transmission. (This is usually used for intercoms, not radios.)
5. Optionally, click Master Push to Talk to transmit on all radios that are not muted.
6. To operate a radio head:
    a. Make sure the radio head power button is enabled ().
    b. Optionally, select a channel from the list.
    c. Click Push to Talk or press **T** to talk (if this is not a Vox radio).
    d. Click the microphone button () to mute the microphone or enable it.
    e. Click the speaker button () to mute the speaker or enable it.
    f. Adjust the slider to change the radio's volume.
    g. Click the Jam button () to jam the radio frequency.
7. Click Configure Radios to edit the radio configuration. See "58.4. Configuring the MAK Radio" on page 1154 for details.
8. In the Radio Communications dialog box only, you can click the audio settings button () to access the Audio Settings for VR-Forces. See "57.2.1 Configuring Sound Effects" on page 1143 for details.

   In the MAK Radio standalone application, you can choose **Settings > Audio Settings** to select the Playback Device (speaker), Recording Device (microphone), and default Push-to-Talk button using the Audio Settings. See "58.5. Audio Settings" on page 1158.

Figure 442. Radio Communications dialog box and standalone application



58.2.3 Operating the Radio from the Radio Control Toolbar

You can operate the radio from the toolbar (Figure 443).

Figure 443. MAK Radio toolbar



➤ **To operate the radio using the toolbar:**

1. Choose a radio head to select it for use.
2. Optionally, select a channel from the list.
3. Click PTT to talk (if this is not a Vox radio).
4. Click the microphone button (🎤) to mute the microphone or enable it.
5. Click the speaker button (🔊) to mute the speaker or enable it.
6. To change the radio's volume, click the arrow button to open and adjust the slider.
7. Click the radio communications button (📻) to open the Radio Communications dialog box (Figure 442 above).

If the toolbar options are disabled, you need to enable radios in VR-Forces. In the Radio Communications dialog box or the standalone application (Figure 442 on the previous page), select the Enable Radio System check box.

58.3. Using the Intercom or a VOX Radio

The intercom and VOX radios are voice activated and transmit any sound greater than the VOX threshold.

➤ **To use the intercom or a Vox radio:**

1. In the Radio Communications dialog box or MAK Radio Application, select the Attached Entity.
2. In the dialog box or application, make sure the power button is enabled ( ).

   In the toolbar, select the intercom or the VOX radio from the list. The PTT button is disabled.
3. Make sure that the microphone is not muted.
4. If you are using a Vox radio, you can select a channel.
5. Start talking.

58.4. Configuring the MAK Radio

You can configure the number of radios and channels, the channel frequencies, the sample rate, and the format. The Modulation and Crypto settings for radios are as defined for the Transmitter PDU in the IEEE DIS standard.

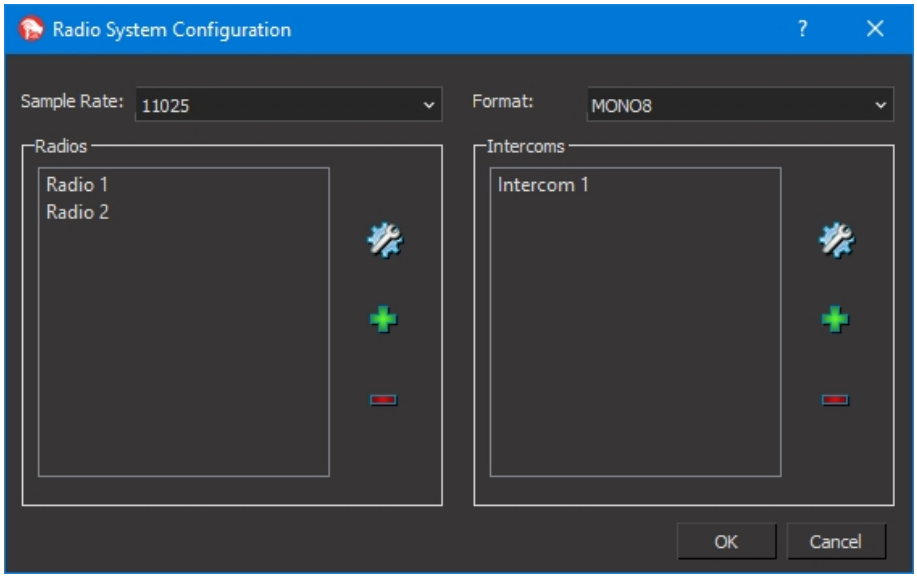
If there is a mismatch between two or more radios in the Transmission, Modulation, and Crypto settings, they cannot talk to each other. The settings must match between communicating radios for the signals to be processed and the transmissions to be heard.

Changes that you make persist from one session of VR-Forces and MAK Radio to another, and the settings are the same for both.

➤ **To configure the MAK Radios:**

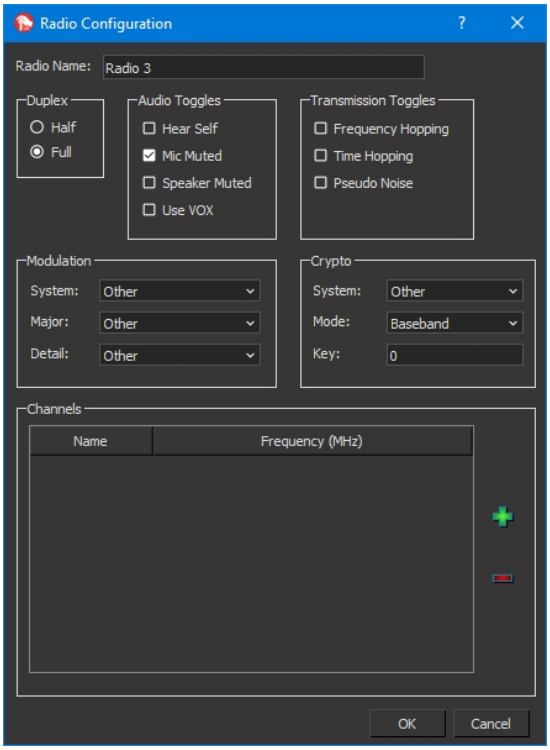
1. In the Radio Communications dialog box or MAK Radio application, click Configure Radios. The Radio System Configuration dialog box opens (Figure 444).

Figure 444. Radio System Configuration



2. To change the sample rate for all radios, select the rate from the list.
3. To change the format for all radios, select the format from the list.
4. To add or edit a radio:
    a. Click the Add Radio ( ) or Edit button ( ) next to the list of radios. The Radio Configuration dialog box opens.
    b. Configure the radio as needed. Table 49 on page 1157 describes the settings.

Figure 445. Radio Configuration



c. Click OK.

5. To add or edit an intercom:

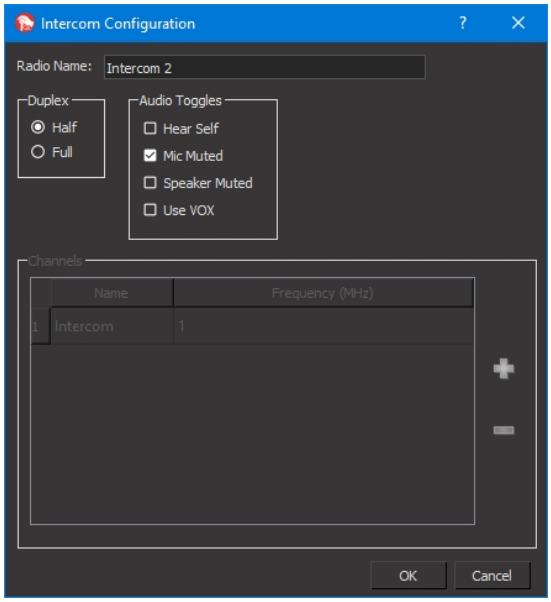
   a. To add an intercom, click the Add Intercom (). A row is added.

   b. To edit a new or existing intercom, select the row and then click the Edit button () next to the intercom list. The Intercom Configuration dialog box opens (Figure 446).

   c. If desired, edit the name of the intercom.

   d. Configure the intercom as needed. Table 49 provides information about the settings. Note that only Duplex and Audio Toggles settings are available for intercoms.

Figure 446. Intercom Configuration



e. Click OK.

6. When you are finished, click OK in the Radio System Configuration dialog box. You return to the Radio Communications dialog box. If you added or removed a radio or intercom, the list updates.

Table 49. Radio and Intercom settings

Setting	Options
Radios and Intercoms	
Duplex	— Half. Only one user can talk at a time. — Full. All users can speak simultaneously.
Audio Toggles	— Hear Self. Users can hear themselves over the speaker. — Mic Muted. Microphone defaults to muted. — Speaker Muted. Speaker defaults to muted. — Use Vox. Radio is voice-activated.
Radios only	
Transmission Toggles	— Frequency Hopping. Switches frequencies during transmission to reduce interference and avoid interception. — Time Hopping. Varies the period and duty cycle to achieve anti-jamming or low probability of intercept. — Pseudo Noise. Introduces a deterministically generated random sequence of bits.

Table 49. Radio and Intercom settings (continued)

Setting	Options
Modulation	— System. Radio system type: Generic, SINGCGARS, and so on. — Major. Modulation type: Amplitude, Pulse, and so on. — Detail. Additional information about the modulation type (options based on the Major setting).
Crypto	— System. Cryptographic system for the radio: VINSON, KY-100, and so on. — Mode. Crypto mode: Baseband, Diphase, and so on. — Key. A simulated cryptographic key. This key is a non-zero integer value for simulating secured communication. Set to 0 when simulating plain communications.
Channels	— Use the Add or Delete buttons to manage the radio's channels. — In the rows, double-click to edit each channel's name and frequency as needed.

58.5. Audio Settings

In the MAK Radio standalone application, you can access the Audio Settings. These settings allow you to specify the Playback Device (speaker), Recording Device (microphone), and one or more push-to-talk buttons.

The options are saved in *./appData/settings/makRadio/radioAudioSettings.csv*. You can copy the CSV file to other workstations that should have the same settings.

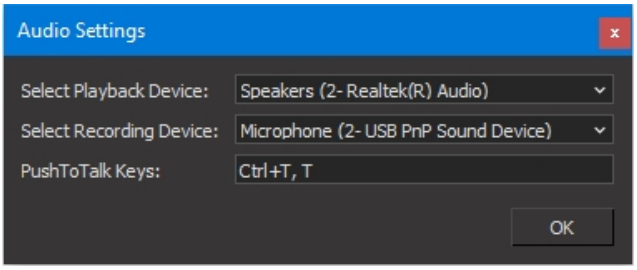
➤ **To choose a Playback Device or Recording Device:**

1. In the MAK Radio standalone application, choose **Settings > Audio Settings**. The Audio Settings dialog box opens.
2. Choose the speaker or microphone from the appropriate list.
3. Click OK.

➤ **To set the default push-to-talk key:**

1. If the Audio Settings dialog box is not open, choose **Settings > Audio Settings**.
2. Click on the Push To Talk Keys box to give it focus.
3. Press the desired key or key combination. If you want to use more than one key, continue pressing keys. Each key or key combination is separated by a comma (Figure 447).
4. Click OK.

Figure 447. Audio Settings for MAK Radio



58.6. Jamming Radios

The Radio Communications dialog box or MAK Radio application allow you to jam radio frequencies. When you click the Jam button on a radio head, it creates another radio that jams the frequency of the currently selected channel. The users will not be able to hear the radio communications, and will not see an indication that their radio is jammed.

➤ To jam a radio frequency, click the Jam button (Jam).



XI. The Observer and Viewing Scenarios

The observer, or eyepoint, is the location in the 3D or 2D environment from which you observe the scene. You can move the observer (navigate) through the scene. You can attach the observer to a simulation object or prop. When the observer is attached to a simulation object, it moves automatically with that object. You can also control the observer from a remote application using view control messages.

You can have multiple observers in a VR-Forces session. They might be associated with different windows. For example, you could have an observer for the main window and one for an inset window.

The observer is such a pervasive component of the VR-Forces GUI that it is covered in multiple places in this manual.

- "Chapter 59. The Observer and Observer Modes" on page 1163
- "Chapter 60. Moving the Observer" on page 1175
- "Chapter 61. Attaching the Observer to Objects" on page 1209
- "59.10. Displaying Models for Remote Observers" on page 1173



59. The Observer and Observer Modes

This chapter explains how to create, edit, and change observers and observer modes. For details about moving the observer, see "Chapter 60. Moving the Observer" on page 1175. For details about attaching the observer to simulation objects and props, see "Chapter 61. Attaching the Observer to Objects" on page 1209.

59.1. Switching Between the 2D and 3D Projections 1164
59.2. Observer Modes 1164
59.3. Changing the Observer Mode 1165
59.4. Creating and Editing Observer Modes 1166
59.4.1 Creating an Observer Mode 1166
59.4.2 Deleting an Observer Mode 1167
59.4.3 Editing an Observer Mode 1167
59.4.4 Renaming an Observer Mode 1167
59.5. Adding an Observer 1167
59.5.1 Adding an Observer on the Observer Settings Page 1168
59.5.2 Adding an Observer in the Display Engine Configuration Editor 1168
59.6. Removing an Observer 1170
59.7. Introduction to XR Mode 1171
59.8. Model Sets 1172
59.8.1 Changing the Model Set 1172
59.9. Communicating Information About the Observer 1172
59.9.1 Making the Observer Visible to Other Applications 1172
59.9.2 Viewing and Attaching to Other Observers 1172
59.10. Displaying Models for Remote Observers 1173

59.1. Switching Between the 2D and 3D Projections

VR-Forces supports both 2D and 3D projections. You can switch between them simply by changing the observer mode from Stealth or Out-the-Window to Plan View. (Changing observer modes is described in "59.3. Changing the Observer Mode" on the facing page.) When you go from a 3D observer mode to 2D, VR-Forces calculates a zoom level so that the observer should appear as if it is roughly the same height above the terrain. When you go from a 2D observer mode to 3D, VR-Forces calculates an altitude to do the same thing, and keeps the 3D view pointing down (so the change in overall view stays fairly small).

59.2. Observer Modes

Observer modes let you configure VR-Forces settings that affect how the observer moves and what it sees. You can then quickly change from one group of settings to another by switching between observer modes. An observer mode specifies:

- A model set. A collection of models of a particular type, such as 3D models.
- A key map. A coordinated set of key mappings to movement functions.
- A projection (2D or 3D).
- Settings for simulation object graphics and observer movement.

VR-Forces includes the following observer modes:

- Stealth. Stealth observer mode is a 3D projection configured for situational awareness.
- Out-the-Window. Out-the-Window observer mode is configured for when you use VR-Forces as a desktop image generator. It hides many of the explanatory visual effects present in the Stealth observer mode.
- Plan View. The Plan View observer mode provides a 2D "big picture" view of the terrain.
- XR. XR, or exaggerated reality, observer mode is a 3D mode that helps you understand the relationships between entities and the terrain when the observer is too far away from entities to view them at realistic sizes. It allows you to exaggerate entity size and terrain.
- IR BlackHot (CameraFX). A view using the CameraFX IR (infrared) BlackHot sensor.
- IR (CameraFX). A view using the CameraFX IR sensor.
- NVG (CameraFX). A view using the CameraFX NVG (night vision goggles) sensor.
- Camera (CameraFX). A view using the Visual Camera CameraFX sensor.

The observer modes all support the three model sets: 3D models, 3D colorized models, and 2D icons. You can edit observer modes and add new ones. For details, see "59.4. Creating and Editing Observer Modes" on page 1166.

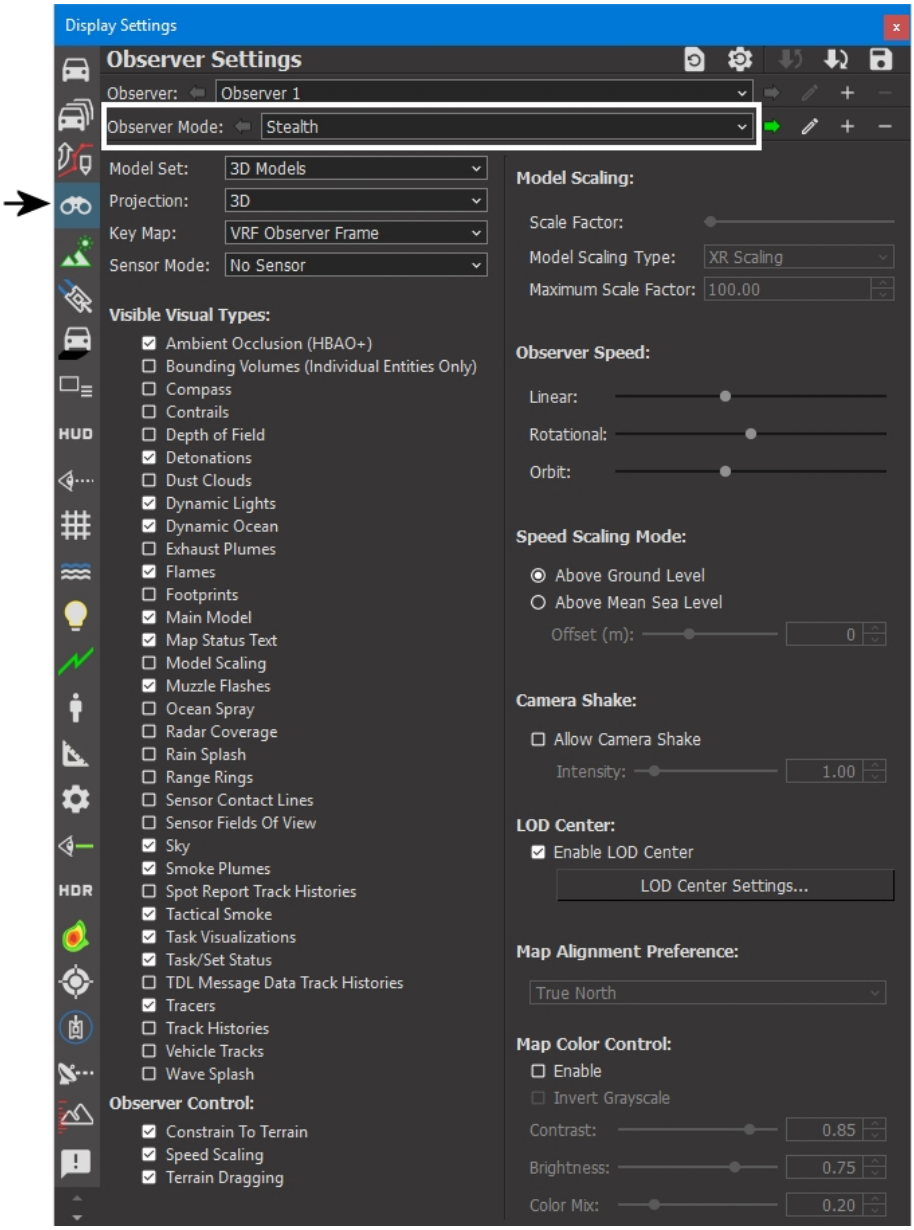
59.3. Changing the Observer Mode

When you change the observer mode, terrain may be displayed differently and the model sets used to display simulation objects may change. For descriptions of the observer modes, see "59.2. Observer Modes" on the previous page.

➤ **To change the observer mode:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448).

Figure 448. Observer Settings page



3. Select an option in the Observer Mode list.

> You can also change the observer mode by choosing **Observer > Set Observer Mode > *mode***, or on the Observer Settings Toolbar, choose an option from the Observer Mode list.

59.4. Creating and Editing Observer Modes

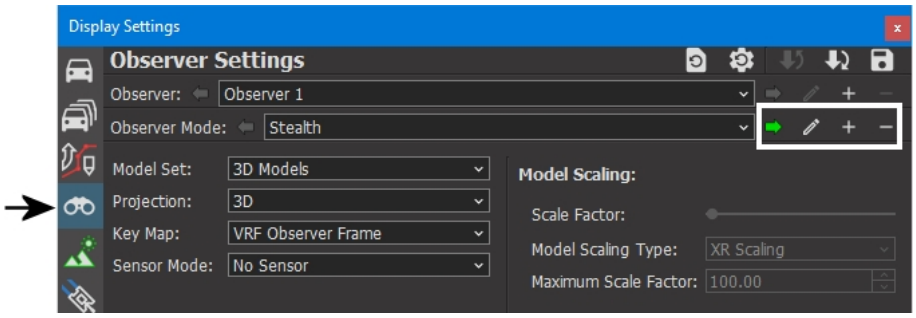
VR-Forces comes with 3D and 2D observer modes. (Table 50 on page 1181 lists the observer modes available for each VR-Forces application.) They are configured with the display settings, key mappings, and model sets that users most commonly want to use. However, if they do not meet your needs, you can easily edit them or create new observer modes.

59.4.1 Creating an Observer Mode

➤ **To create an observer mode:**

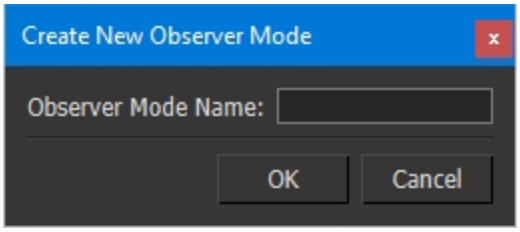
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449).

Figure 449. Observer Settings page



3. Click the Add button (+) to the right of the Observer Mode list. The Create New Observer Mode dialog box opens (Figure 450).

Figure 450. Create New Observer Mode dialog box



4. Type a name for the new observer mode.
5. Click OK.
6. In the Model Set list, select a model set.
7. Select a projection (2D or 3D).

8. Select the key map you want to use.
9. Optionally, specify a sensor mode.
10. Optionally, change the observer speed scaling settings and speed scaling mode.
11. Optionally, for 3D model sets, change the Model Scaling settings.
12. Enable or disable the mode-specific settings. The list of settings varies depending on the projection and model set.

59.4.2 Deleting an Observer Mode

➤ **To delete an observer mode:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449 on the previous page).
3. In the Observer Mode list, select the observer mode that you want to delete.
4. Click the Delete button ( ).

59.4.3 Editing an Observer Mode

➤ **To edit an observer mode:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449 on the previous page).
3. In the Observer Mode list, select the observer mode that you want to edit.
4. Edit the attributes of the observer mode as desired.

59.4.4 Renaming an Observer Mode

➤ **To rename an observer mode:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449 on the previous page).
3. In the Observer Mode list, select the observer mode that you want to rename.
4. Click the Rename button ( ). The Rename Observer Mode dialog box opens.
5. Type the new name.
6. Click OK.

59.5. Adding an Observer

You can add and remove observers. Having multiple observers lets you assign different observers to different windows or channels. Changing the position and orientation of the observer in one channel does not affect the view of a different observer in a different channel.

> If you have multiple observers, the observer listed in the Observer Settings Toolbar is the observer assigned to the window that has focus. If you open the Display Engine Configuration Editor Panel and select a different observer in the Channel Attributes list, the data in relevant panels will change to reflect the values for the selected observer. However, as soon as you return focus to a window, the observer selection will change to match that of the window.

You can add observers in the Display Engine Configuration Editor Panel or on the Observer Settings page.

If you want to save your observers, save the display engine configuration in which you created them. (Saving an observer just saves its name. It does not save any information about a particular observer mode or location that you may have used for that observer. To save location information, save an observer view.)

59.5.1 Adding an Observer on the Observer Settings Page

➤ **To add an observer on the Observer Settings page:**

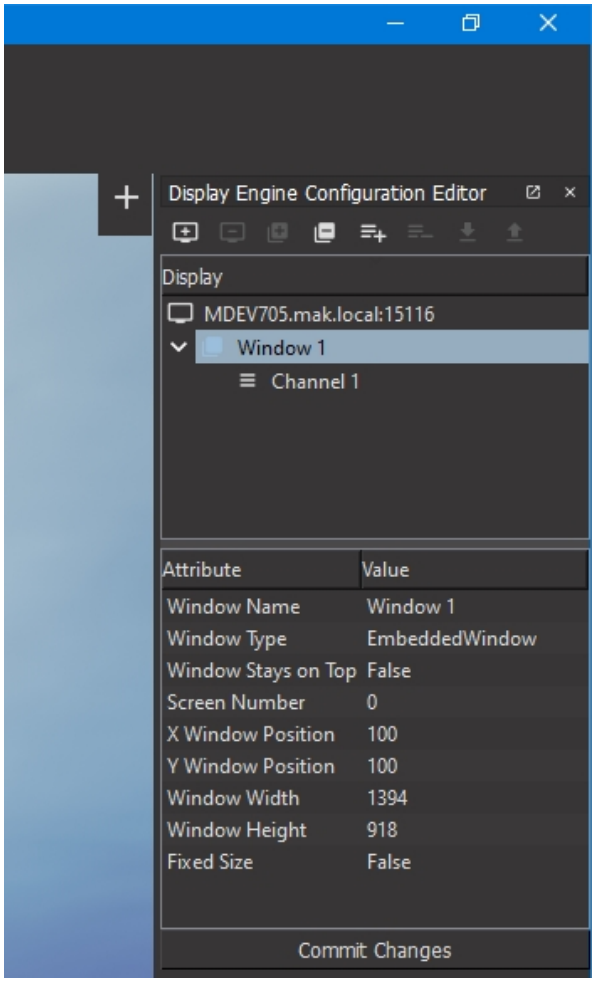
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449 on page 1166).
3. Click the Add button (+) to the right of the Observer list. The Create New Observer dialog box opens.
4. Type a name for the new observer.
5. Click OK. The new observer is added to the list.
6. To save the observer, save the display configuration. (Saving the observer settings does not save observers. For information saving a display configuration, see "83.2.5 Saving a Display Engine Configuration" on page 1602.)

59.5.2 Adding an Observer in the Display Engine Configuration Editor

➤ **To add an observer on the Display Engine Configuration Editor Panel:**

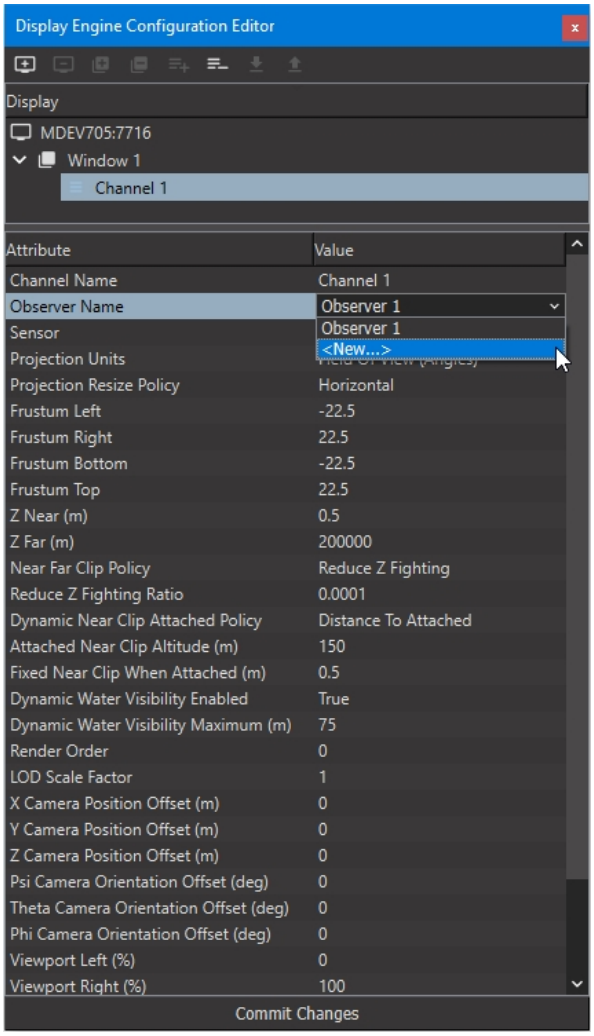
1. Choose **View > Display Engine Configuration Editor Panel**. The Display Engine Configuration Editor Panel opens (Figure 451).

Figure 451. Display Engine Configuration Editor Panel



2. Select the channel on which you want to add an observer.
3. Click the Value field for the Observer Name attribute to make it editable.
4. In the list, select New (Figure 452).

Figure 452. Observer Name attribute



5. Click Commit Changes. The Add Observer dialog box opens.
6. Type a name for the new observer.
7. Click OK. The new observer becomes the selected observer on the Observer Settings Toolbar, and is listed in the Observer Name attribute. (You may need to refresh the channel attributes before the observer name is listed.)
8. To save the observer, save the display configuration. (For information about saving a display configuration, see "83.2.5 Saving a Display Engine Configuration" on page 1602.)

59.6. Removing an Observer

When you close VR-Forces, all observers are removed. They are not saved unless you save a display configuration. You can also remove observers during a session.

➤ **To remove an observer:**

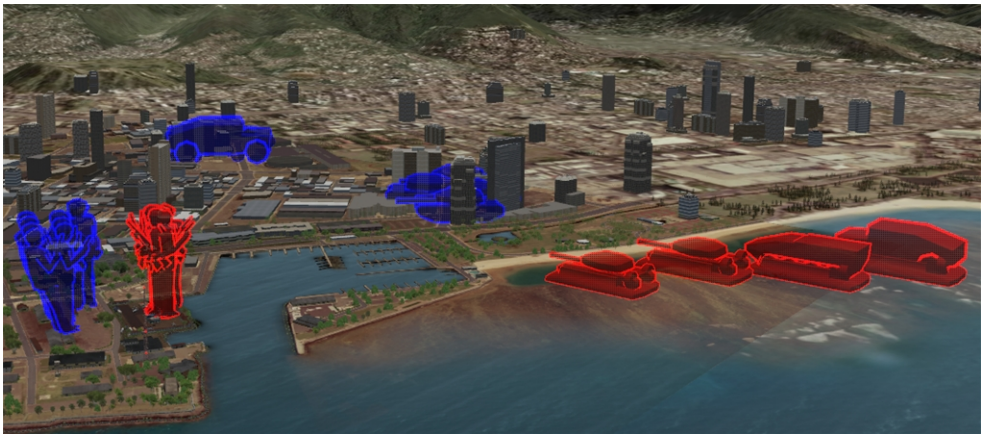
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 449 on page 1166).
3. In the Observer list, select the observer that you want to remove.
4. Click the Delete button (—). The observer is removed.
5. If the observer was saved as part of a display configuration, save the updated display configuration.

59.7. Introduction to XR Mode

The Stealth observer mode (3D projection and 3D model set) provides situational awareness at the entity or small unit level. It is not ideal for getting the big picture of a simulation, because when you move the observer out to see large areas of terrain, the entity models, which are scaled to the terrain, are too small to see. The Plan View observer mode provides the big picture of a simulation, but does not provide situational awareness.

XR observer mode bridges the gap between the Stealth and Plan View observer modes. It is a 3D mode, so it provides situational awareness. However, it uses a special model set that makes models visible at greater distances than in Stealth observer mode. The 3D colorized model set shows entities as notional models using force coloring, which helps them stand out from the terrain. It also scales the model size to provided an exaggerated reality (hence XR). (You can also scale the regular 3D model set if you want to.) Figure 453 illustrates model scaling with 3D colorized models.

Figure 453. Model scaling and colorized models



For details about scaling models, see "51.2. Exaggerating the Scale of 3D Entity Models" on page 1082.

59.8. Model Sets

VR-Forces includes these model sets:

- 3D Models. Realistic, 3D, textured models with articulated parts.
- 3D Colorized Models. 3D model caricatures with minimal articulation. These models do not have textures, they are solid colors, by force. You can change the coloring between primary colors and MIL-STD icon colors.
- 2D icons. MIL-STD 2525B icons.

You can use any of the model sets in both 3D and 2D views. In the 2D view, the 3D models can only be viewed from the top.

59.8.1 Changing the Model Set

Each observer mode specifies a model set. You can change the model set associated with an observer mode by editing it (for details, see "59.4.3 Editing an Observer Mode" on page 1167), but you can also dynamically change the model set being used.

➤ To change the model set, choose **Observer > Set Model Set > *model_set***, where *model_set* is one of the options on the menu.

59.9. Communicating Information About the Observer

VR-Forces can transmit state updates describing the observer's position and orientation, and can respond to view control messages sent by other applications.

59.9.1 Making the Observer Visible to Other Applications

By default, VR-Forces does not send state update messages to make its state information known to other applications. When you publish observer information, a plan view display can represent VR-Forces as an icon, or one VR-Forces application can synchronize its observer movement with that of another VR-Forces application.

➤ To enable or disable broadcasting of observer information, choose **Settings > Publish Observers** (or **Stop Publishing Observers**).

Making the Observer Visible to Only One Application (DIS Only)

In DIS you can send observer information to just one other application, rather than to everyone in the exercise. To do this, in the Edit Connection dialog box for the DIS connection, instead of using a broadcast address, set the destination address to the address of a specific computer.

59.9.2 Viewing and Attaching to Other Observers

If an exercise includes other stealth applications and they are sending state updates, VR-Forces can detect them. A remote observer entity is listed in the object hierarchy, so you can attach the local observer to it just as you would attach to any other simulation object.

Observer icons are displayed in the 2D view, but they are not normally visible in the 3D view, even when they appear in the entity list. If you want to display visual representations of remote observers, map a visual model to the observer's entity enumeration.

For related information, see "59.10. Displaying Models for Remote Observers" below.

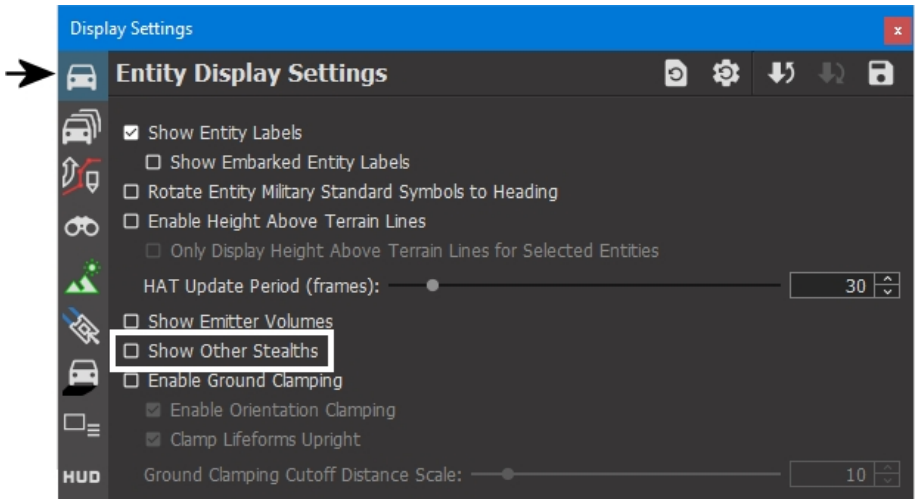
59.10. Displaying Models for Remote Observers

VR-Forces broadcasts the location and orientation of the observer. By default, it does not display a model for the observer. However, if there are other applications in an exercise that are broadcasting a stealth PDU (such as another VR-Forces application), you might want to represent them with a model.

➤ **To display a model for remote observers:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 454).

Figure 454. Display Settings, Entity Display Settings page



3. Select the Show Other Stealths check box.



60. Moving the Observer

This chapter explains how to move the observer (eyepoint) in the VR-Forces window.

60.1. Moving the Observer 1177
60.1.1 Using the Compass to View the Observer's Heading 1178
60.1.2 Setting the Compass to Degrees or MILs 1179
60.1.3 Editing the Compass's Attributes 1180
60.2. Observer Movement Functions and Keyboard Mappings 1181
60.2.1 3D Observer Movement Coordinate Systems 1182
60.2.2 2D Observer Coordinate System 1182
60.2.3 The Attached Context and Observer Movement (3D Only) 1182
60.3. Moving the Observer from the Keyboard in the 3D View 1184
60.3.1 Navigation Options 1185
60.3.2 Orbiting 1186
60.3.3 Moving the Camera 1186
60.4. Moving the Observer Using the Mouse (3D View) 1187
60.4.1 Dragging the Terrain 1187
60.4.2 Changing the Observer's Orientation with the Mouse 1189
60.4.3 Dragging the View 1189
60.4.4 Orbiting the Terrain with the Mouse 1189
60.4.5 Teleporting the Observer 1189
60.5. Moving the Observer in the 2D View 1190
60.5.1 Moving the Observer from the Keyboard in the 2D View 1190
60.5.2 Moving the Observer Using the Mouse in the 2D View 1191
60.6. Zooming the Observer 1191
60.6.1 Zooming the Observer in the 3D Projection 1191
60.6.2 Zooming the Observer in the 2D Projection 1192
60.6.3 Setting the Scale Factor for Terrain 1192
60.6.4 Setting the Magnification LOD Angle for Terrains 1193
60.6.5 Zooming In to the Ocean Over Long Distances 1194

60.6.6 Zooming to Extents 1194
60.6.7 Configuring the LOD Center in the Observer Mode 1195
60.7. Centering the Observer on a Simulation Object 1197
60.7.1 Centering on an Object from the Object Console Summary Panel 1197
60.7.2 Centering on an Object from an Information Panel 1198
60.8. Simulating Camera Jitter 1198
60.9. Forcing Orientation North 1199
60.9.1 Specifying True, Magnetic, or Grid North 1199
60.10. Enabling and Disabling View Constraints (3D Only) 1200
60.11. Controlling Navigation Speed 1200
60.11.1 Enabling or Disabling Speed Scaling 1200
60.11.2 Changing the Observer's Movement Speed 1201
60.12. Saving and Recalling Views 1202
60.12.1 Recalling a View 1204
60.12.2 Deleting Views 1204
60.12.3 Loading Views Automatically When You Load a Terrain 1204
60.12.4 Specifying the Scenario Startup View 1205
60.12.5 Saving Views to a File 1206
60.12.6 Loading Views 1206
60.13. Using the Go To Location Panel 1207
60.14. Controlling the Observer from Other Applications 1207
60.14.1 Enabling the Processing of View Control Messages 1208

60.1. Moving the Observer

When you view a simulation, you usually want to move the observer (eyepoint) to different locations to view the terrain and scene elements being simulated. In VR-Forces, you can move the observer using the keyboard and the mouse. There are two basic contexts in which you can move the observer:

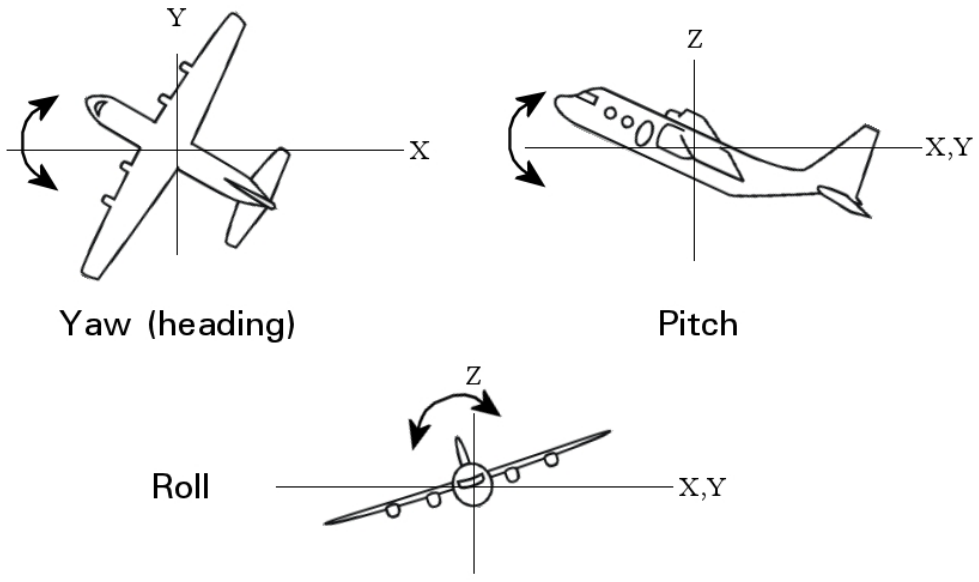
- You can move it independently of any scene element. This is called absolute, or free-fly mode.
- You can attach the observer to a simulation object, in which case it moves with the object. (You can also attach the observer to a prop. For details about attaching the observer to a simulation object or prop, see "Chapter 61. Attaching the Observer to Objects" on page 1209.)

Absolute mode and the various attachment types are collectively called attach modes.

Regardless of which attach mode you are using, you can change the position and orientation of the observer. In the 3D view, position is defined by X, Y, and Z coordinates that identify its location in the 3D world. Orientation describes the observer's yaw (heading), pitch, and roll (Figure 455).

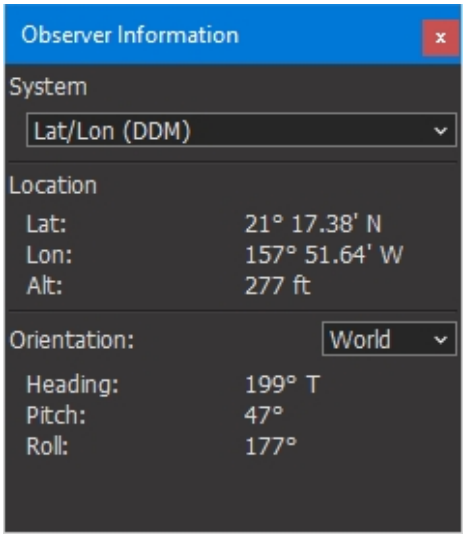
In the 2D view, the Z coordinate is fixed. You can change the X and Y coordinates. You can change the observer's yaw, but pitch and roll are fixed.

Figure 455. Orientation



The Observer Information Panel (Figure 456) displays the location and orientation of the observer in whichever coordinate system you select.

Figure 456. Observer Information Panel



60.1.1 Using the Compass to View the Observer's Heading

VR-Forces can display a compass that always shows the heading of the observer when a terrain is loaded. The compass is an observer mode setting. In Figure 458, the observer's heading is east.

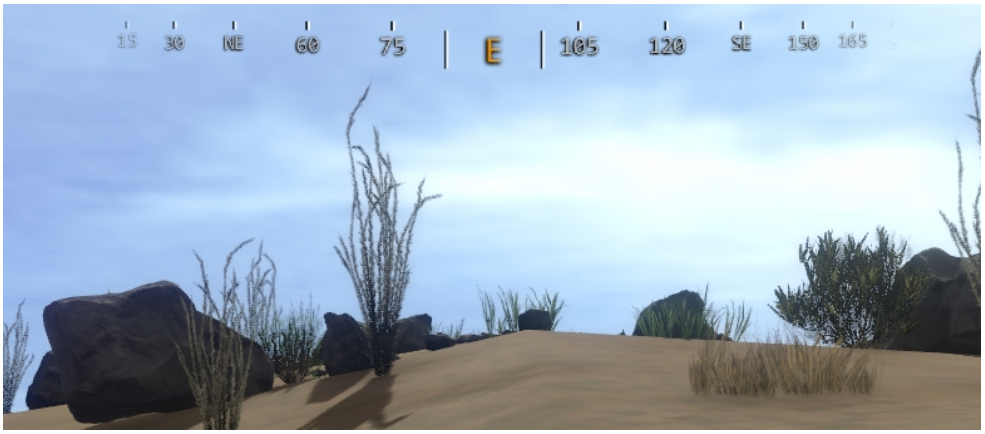
In the 2D view, the compass has two needles. The straight needle indicates true north and the arrow needle indicates magnetic north. Depending on where the observer is in the world, the needles may be close together or wide apart.

Figure 457. 2D Compass



In the 3D view, the compass is horizontal and shows the direction where the observer is looking. This uses either true or magnetic north, based on the Heading specified in the Display Units Settings. You can also specify whether the 3D compass should use degrees or NATO-MIL. See "84.9. Specifying Display Units" on page 1634.

Figure 458. 3D Compass



➤ **To enable or disable the compass:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448 on page 1165).
3. Select the observer mode for which you want to change the compass setting.
4. In the list of observer settings, select or clear Compass.

You can also enable or disable the compass by choosing **Observer > Compass**.

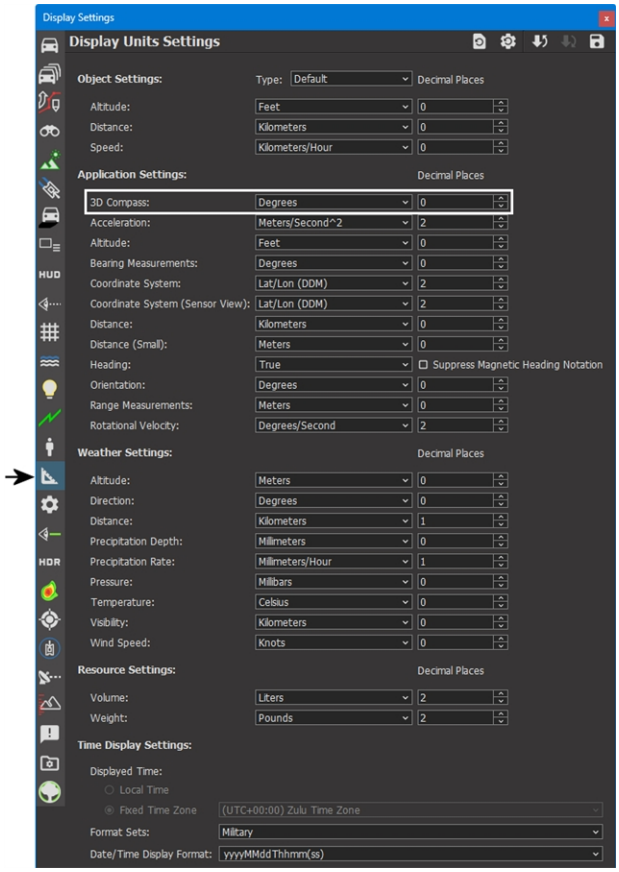
60.1.2 Setting the Compass to Degrees or MILs

In the Display Units Settings, you can specify the measurement units for the 3D compass: degrees or NATO-MIL.

➤ **To specify the measurement units for the 3D compass:**

1. Choose **Settings > Display**.
2. Select the Display Units Settings page.
3. From the 3D Compass list, choose the measurement you want to use.

Figure 459. 3D Compass Settings



When you use the 3D compass, the default behavior is for it to show the cardinal and intercardinal direction indicators (N, NE, S, SW, and so on). You can change the definition such that they do not display as the current heading. (They continue to display on other parts of the compass.)

If you want to change the setting, edit the 3dViewCompass model definition (in the Channel3dViewCompass schema) and set the parameter isExactHeadingAlwaysShown to true. For more information, see the *Model Definitions* chapter of the *Adding Content to MAK ONE Applications Reference Manual*.

60.1.3 Editing the Compass's Attributes

You can configure the compass by editing its schema (ChannelCompass) and model definition (smallCompass and largeCompass). (For details about editing schemas, see "Creating and Editing Schemas" in the *Adding Content to MAK ONE Applications Reference Manual*. For details about editing model definitions, see "Creating and Editing Model Definitions" in the *Adding Content to MAK ONE Applications Reference Manual*.)

By default, all channels have the largeCompass keyword. To change the style of compass used for a channel, change the keyword in the Keywords attribute for the channel. To completely remove the compass, delete the keyword. For details about setting channel attributes, see "83.4. Changing a Channel's Attributes" on page 1604.

You can change where the compass displays in the window by editing the location attribute. The permissible values are top, topRight, right, bottomRight, bottom, bottomLeft, left, and topLeft.

You can change the color of the compass by editing the darkTickColor and lightTickColor attributes. The color is defined using the OpenGL scheme for specifying RGBA. The first three values are the RGB color values. The fourth value specifies opacity. Each value can be from 0.0 through 1.0.

60.2. Observer Movement Functions and Keyboard Mappings

When you move the observer, you are executing observer movement functions. An observer movement function is a specific way that the observer can move, such as Move Observer Forward or Move Level Observer Forward. (These functions both move the observer forward, but they use different observer coordinate systems. Observer movement coordinate systems are described in "60.2.1 3D Observer Movement Coordinate Systems" on the next page.)

The various movement functions are mapped to the keys on the keyboard and to mouse buttons and the mouse wheel. If you use the default key mappings for the default observer modes, then most of the time, the observer will behave exactly as you would expect it to. However for some types of movement, such as forward and back, the observer can move differently depending on the view, the observer coordinate system, or the attach mode. Therefore, VR-Forces provides several key mappings that let you choose which type of observer movement to use. You can change these key mappings and create your own key mappings. (For details about editing key maps, see "Chapter 86. Creating and Editing Key Mappings" on page 1649.) This section describes how observer movement can vary based on the observer coordinate system and the attach mode.

Table 50 lists the observer modes and key map settings provided with VR-Forces applications.

Table 50. Observer modes

Observer Mode	Projection	Key Map
Stealth	3D	Observer Frame
CameraFX modes	3D	Observer Frame
XR	3D	Observer Frame
Plan View	2D	2D Level Observer Frame
The observer modes all support the three model sets: 3D models, 3D colorized models, and 2D icons.		

60.2.1 3D Observer Movement Coordinate Systems

In the 3D view, the observer moves in the context of one of the following coordinate systems:

- Level Observer Coordinates: Movement is with respect to the terrain. The level observer coordinate system is defined by the current up vector (Z), and a plane normal to that vector into which the observer's X and Y axes are projected. (Essentially, this means that the observer's heading is used while its pitch and roll are considered to be 0.0.)

> Movement is always with respect to a plane tangent to the earth's surface. However, in a projected terrain database, such as UTM, Lambert Conformal Conic, or flat earth, the plane is fixed, whereas in a geocentric terrain database the up vector (and hence the tangent plan) changes as the observer's position changes, so movement follows the curvature of the earth.

- Observer Coordinates: Movement is with respect to the observer's body coordinate system.
- Entity Level Body Frame Coordinates: Movement is with respect to an entity's body frame, but ignores pitch and roll.
- Entity Body Frame Coordinates: Movement is with respect to an entity's body frame, including pitch and roll.

The practical difference between observer coordinates and level observer coordinates is that in level observer coordinates, if you are looking down at the terrain, when you move forward and backward, you fly over the terrain, looking down. In observer coordinates, when you move forward and backward, you move down towards the terrain or away from it at whatever angle you are oriented towards it.

60.2.2 2D Observer Coordinate System

When you use the 2D view in VR-Forces, the observer uses a variation of level observer coordinates. Movement is restricted to two dimensions, so there are no movement functions for changing the altitude, pitch, or roll. You can zoom in and out in the 2D view, but this does not change the observer's altitude, it changes the field of view and resolution of the terrain image.

60.2.3 The Attached Context and Observer Movement (3D Only)

The key maps control navigation using a particular coordinate system (observer frame or level observer frame). They also support navigation in alternate coordinate systems. The alternate system varies depending on the attached context of the observer. It represents MAK's opinion of which alternate coordinate system would be most useful given the attach mode that the observer is using.

The Move Attached Context *direction* (forward, back, up, down, left, right) functions move the observer in the context of its attach mode. Movement is modified by attach modes:

- Follow mode: movement is in the entity's level body frame.
- Track mode: movement is in the entity's level body frame.
- Mimic mode: movement is in the entity's body frame.
- Tether mode: movement is in level observer frame.
- Unattached: movement is in level observer coordinates.

Figure 460 and Figure 461 illustrate the differences between moving the observer in an attached context and without an attached context.

Figure 460 illustrates movement in entity level body coordinates (Follow and Track modes). The Move Observer Left function (**A** in Observer Frame key map) moves the observer to the left in observer coordinates, that is left relative to the observer's heading. The Move Attached Context Left function (**Shift+A**) moves the observer left in the attached context. In entity level body frame, this means the observer moves left relative to the entity's heading.

Figure 460. Observer movement in entity level body frame coordinates

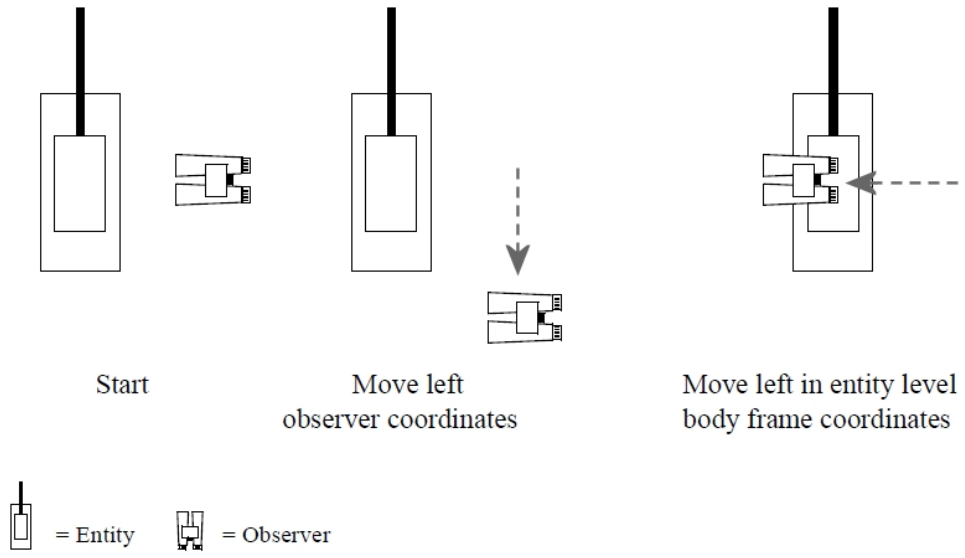
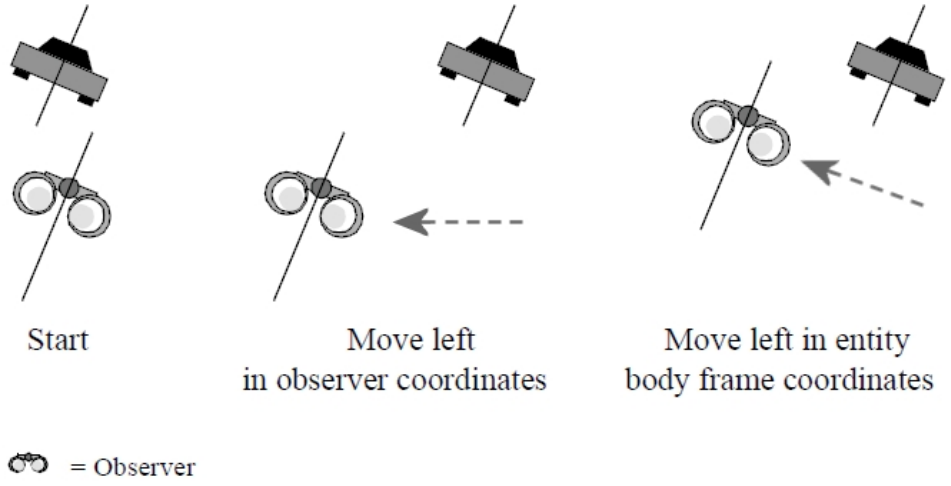


Figure 461 illustrates movement in entity body frame (Mimic mode). The Move Observer Left function moves the observer to the left in observer coordinates, that is left relative to the observer's heading, but ignoring pitch and roll. The Move Attached Context Left function moves the observer left in the attached context. In entity body frame, this means the observer moves left relative to the entity's heading and its pitch and roll.

Figure 461. Movement in entity body frame coordinates



60.3. Moving the Observer from the Keyboard in the 3D View

VR-Forces maps the observer movement functions to the keyboard using key maps that support the most typical ways that users want to move through the 3D world, either in absolute mode or attached to a simulation object.

> ⚠ References to movement key combinations use the key mappings provided with VR-Forces. If you change these key mappings, the instructions may no longer be valid.

As described in "60.1. Moving the Observer" on page 1177, the observer typically moves with respect to the terrain in level observer coordinates or observer coordinates. The corresponding key maps are Level Observer Frame and Observer Frame.

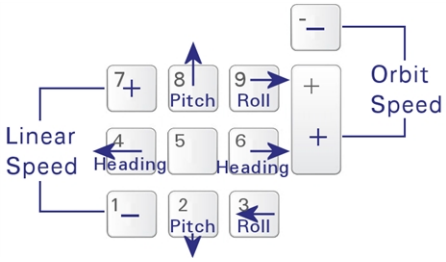
Figure 462 illustrates the positional movement keys. Figure 463 illustrates the keypad layout for changing orientation and movement speed. For a complete list of movement functions and their keyboard and mouse mappings, see the VR-Forces Quick Reference Card or the Key Mapping Editor.

> ⚠ To use the keypad for navigation, NumLock must be on.

Figure 462. Primary movement keys, observer coordinates (3D)



Figure 463. Keypad layout, heading, pitch, and roll (3D)



If your keyboard does not have a keypad, as on most laptops, the Laptop key map lets you use the arrow keys to control the left, right, up, and down movement mapped to 4, 6, 8, and 2 on keypads.

➤ To move the observer, press a key combination and hold it for as long as you want the observer to keep moving.

You can press more than one key at a time to combine movement functions. For example, pressing the move forward and move left keys at the same time moves the observer diagonally left and forward.

60.3.1 Navigation Options

This section lists the different ways that you can move the observer using the keyboard. For a complete list of movement functions and their keyboard and mouse mappings, see the VR-Forces Quick Reference Card. You can:

- Move forward and backward (observer, level observer, entity, level entity, and attached context coordinates).
- Move up and down (observer, level observer, entity, level entity, and attached context coordinates).
- Move left and right (observer, level observer, entity, level entity, and attached context coordinates).
- Rotate (heading) left and right (observer and level observer coordinates).
- Rotate (pitch) up and down (observer and level observer coordinates).
- Rotate (roll) left and right (observer and level observer coordinates).
- Orbit left and right.
- Orbit up and down.
- Look (move camera) left and right.
- Look (move camera) up and down.
- Tilt (move camera) left and right.
- Reset the observer to the default position and orientation for the attached mode.

- Reset the camera to the default position and orientation.
- Zoom the view.

For details about orbiting, see "60.3.2 Orbiting" below. For details about moving the camera, see "60.3.3 Moving the Camera" below.

When you view a geocentric terrain and move the observer more than 6 million meters away from the earth's surface, the view is forced to focus toward the center of the earth. The practical effect of this mode is that the left and right movement keys orbit the eyepoint around the earth, rather than moving the entire earth to the left or right.

60.3.2 Orbiting

You can orbit the observer (revolve left, right, up, and down) around an object or the point on the terrain that is at the center of the window. (If the center of the screen is in the sky, orbit does not work.) When you orbit, the observer's heading continuously changes so that it always faces the point or object around which it orbits. You can change the camera to look at something other than the orbit point while you are orbiting.

➤ **To orbit around an object:**

1. Attach the observer to the object.
2. Press the right or left arrow keys.

➤ **To orbit around a point:**

1. If you are attached to a simulation object, detach (press the **Z** key).
2. Press the right or left arrow keys.

60.3.3 Moving the Camera

By default, the observer always looks in the same direction as its heading. However, just as the turret on a tank can swivel and fire in directions other than its body's heading, the observer can look in a direction other than its heading. The ability to look around is enabled in the camera. You can move the observer's camera up, down, left, right, tilt left, and tilt right. (The default key mappings are **Ctrl**+*Keypad number*, where *number* is 2, 3, 4, 6, 8, or 9.)

To use the keypad for navigation, **NumLock** must be on.

You can tell that the camera is moving, but the heading is not, by observing the compass. If you change the heading, the view of the terrain changes and the compass moves. If you move the camera, but keep the heading static, the view of the terrain changes, but the compass does not move.

➤ To move the camera, press the keys mapped to the camera movement functions.

60.4. Moving the Observer Using the Mouse (3D View)

You can change the observer's position and orientation with the mouse. (To see the full list of mouse mappings, view the Mouse Settings page on the Application Settings dialog box.)

Using the mouse, you can:

- Drag the terrain
- Drag the view
- Orbit the terrain
- Change orientation
- Teleport (jump)

60.4.1 Dragging the Terrain

Dragging the terrain combines the move forward, backward, left, and right functions.

➤ **To drag the terrain:**

1. Click a point on the terrain and hold down the left mouse button.
2. Drag the mouse left, right, forward, and backward.

ⓘ Dragging the terrain is most effective when you are moving large distances, such as several kilometers. To move small distances, use the keyboard.

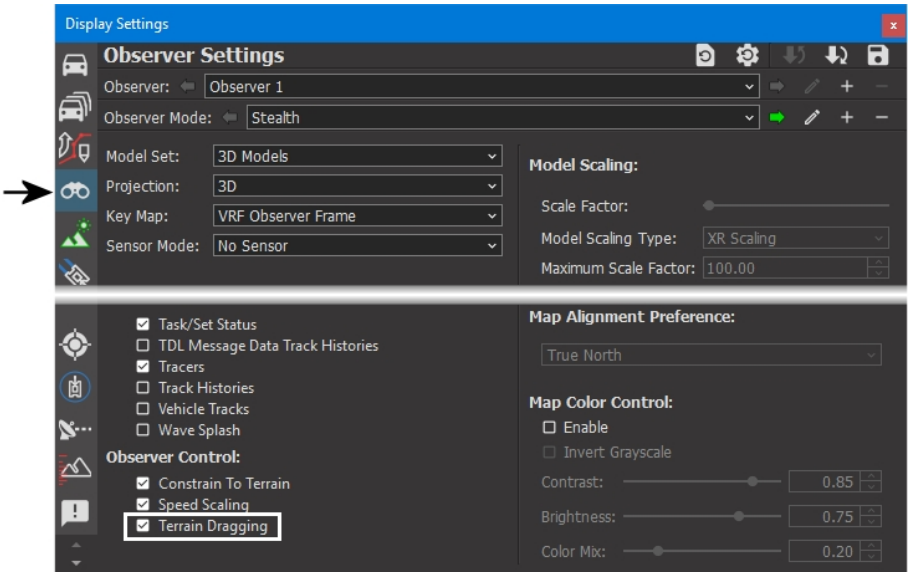
Disabling Terrain Dragging

There may be times when you want to disable terrain dragging. This may be particularly useful when you are creating simulation objects or objects or editing them and do not want to inadvertently drag the terrain when you are trying to move objects.

➤ **To enable or disable terrain dragging:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 464).

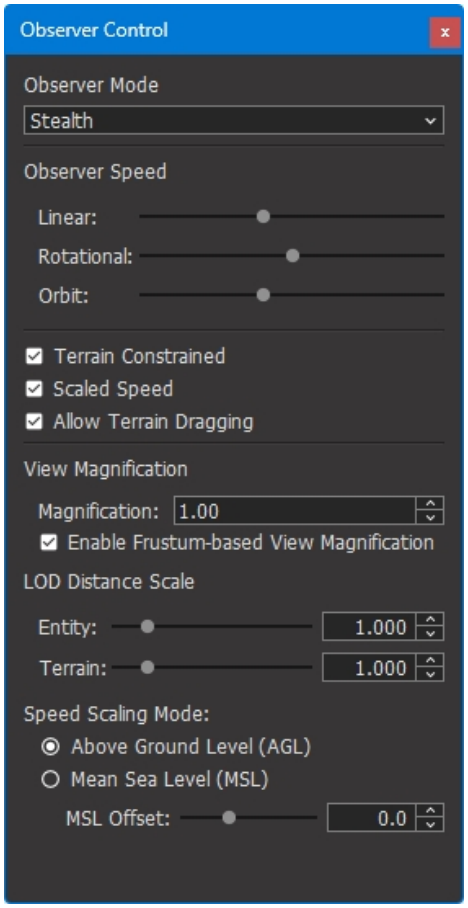
Figure 464. Observer Settings page



3. Select or clear the Terrain Dragging check box.

You can also enable or disable terrain dragging by clearing the Allow Terrain Dragging check box on the Observer Control Panel (Figure 465).

Figure 465. Observer Control Panel



60.4.2 Changing the Observer's Orientation with the Mouse

➤ To change the observer's orientation, hold down the right mouse button and drag the mouse. The observer rotates in the direction you drag the mouse.

60.4.3 Dragging the View

Dragging the view combines the observer up, down, left, and right movement functions. Unlike dragging the terrain, which requires you to click a point on the terrain, you can drag the view from any observer location.

➤ To drag the view, hold down the middle mouse button (or wheel) and drag the mouse.

60.4.4 Orbiting the Terrain with the Mouse

➤ To orbit the terrain with the mouse, press **Ctrl** and drag the mouse.

60.4.5 Teleporting the Observer

You can teleport the observer to a simulation object, a prop, or a place on the terrain. Teleporting moves the observer 90% of the way from its current position to the location or object clicked on.

➤ **To teleport the observer:**

- **Shift**+click on the terrain or prop that you want to jump to.

  If you **Shift**+click on a filled area, the observer attaches to the area; it does not teleport. If you want to teleport to the terrain under an area, hide the area first. You can hide an area by hiding all tactical graphics or hiding the overlay on which the area is located.

- **Ctrl**+**Shift**+click on the simulation object that you want to jump to.

60.5. Moving the Observer in the 2D View

In the 2D view, you move the observer using the mouse and keyboard just as you do in the 3D view. Many of the key and button mappings for the 2D view are the same as those in the 3D view. However, the 2D mappings do not support movement that is not meaningful in two dimensions, such as up and down (altitude), pitch, and roll.

When Orient North is enabled, you cannot change the heading of the observer. For information about Orient North, see "60.9. Forcing Orientation North" on page 1199.

60.5.1 Moving the Observer from the Keyboard in the 2D View

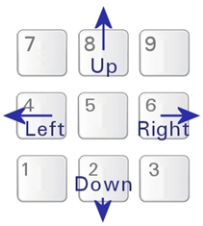
Figure 466 illustrates the positional movement keys for the 2D view. Figure 467 illustrates the keypad layout for changing heading and movement speed. For a complete list of movement functions and their keyboard and mouse mappings, see the VR-Forces Quick Reference Card.

To use the keypad for navigation, **NumLock** must be on.

Figure 466. Primary movement keys, observer coordinates (2D)



Figure 467. Keypad movement (2D)



60.5.2 Moving the Observer Using the Mouse in the 2D View

You can change the observer's position and heading with the mouse. You can change the observer's position by dragging the terrain, as described in "60.4.1 Dragging the Terrain" on page 1187. You can change heading by orbiting, as described in the next section. (To see the full list of mouse mappings, view the Mouse Settings page on the Application Settings dialog box.)

Orbiting the Terrain with the Mouse

When Orient North is enabled, you cannot orbit. For information about Orient North, see "60.9. Forcing Orientation North" on page 1199.

➤ To orbit the terrain with the mouse, press the right mouse button and drag the mouse.

60.6. Zooming the Observer

You can zoom the observer in the 3D and 2D projections. Although the apparent effect is similar for top-down views, they are conceptually and behaviorally different. In addition to a straight "in-and-out" zoom, you can zoom the view to the extents of a simulation object or group of simulation objects.

60.6.1 Zooming the Observer in the 3D Projection

If you want to see something at a higher level of detail (LOD) without moving the observer, you can adjust the view magnification (zoom) of the observer view. Zooming the observer magnifies objects in the view, increasing the LOD to match the level of zoom. The magnification level is saved in saved views. 3D zoom has these characteristics:

- When you zoom the observer view, its movement and mouse look speeds decrease as the magnification increases, to allow for increased control.
- The channel's (window's) field of view establishes the baseline 1X zoom level. Changing the field of view does not change the effective magnification (that is, it does not change the LOD).
- When you switch from 3D to 2D (Stealth only), VR-Forces calculates a 2D zoom level to match the 3D view. If you then switch back to 3D, it remembers the observer location and zoom level.
- If you zoom in from a very high altitude there may be scene corruption due to insufficient precision in the z-buffer. This can be limited by moving the observer closer to the subject before zooming in. This issue is typically only encountered above 10,000 meters.
- When you zoom, VR-Forces respects the environmental conditions at the observer's location. For example, if the observer is above the cloud layer, zooming does not let you see below the clouds. Or, if it is raining at the observer's location, the visibility will not improve as you zoom in on a distant object.
- After zooming in, when you zoom out, you can only zoom out to magnification level 1. That is, you can only zoom out back to the observer's location. To see more of the terrain, move the observer away from the terrain.

➤ To zoom in, press the **=** key, press **Ctrl** and scroll the mouse wheel forward, or on the Observer Control Panel (Figure 465 on page 1189) specify a value in the Magnification box.

➤ To zoom out, press the **-** (dash) key, press **Ctrl** and scroll the mouse wheel backward, or on the Observer Control Panel specify a value in the Magnification box.

Each press of the **=** or **-** key, or increment of the mouse wheel, increases or decreases magnification by a factor of 1.25.

➤ To reset magnification to 1.0, press zero (**0**) or press the middle mouse button.

60.6.2 Zooming the Observer in the 2D Projection

In the 2D projection you can zoom the observer view closer to and away from the terrain. However, unlike in 3D, in which you cannot zoom out past the location of the observer, there is essentially no limit on zooming in and out because in 2D, the observer does not actually have altitude.

➤ To zoom in, press **E**, or to step the zoom, press the **=** key.

➤ To zoom out, press **Q**, or to step the zoom, press the **-** (dash) key.

➤ To zoom with the mouse wheel, scroll forward to zoom in on the spot under the mouse cursor, or backward to zoom out. Hold down the **Alt** key and scroll forward to zoom in on the spot at the center of the screen.

➤ To reset the observer to the default zoom level, press the **Spacebar**.

60.6.3 Setting the Scale Factor for Terrain

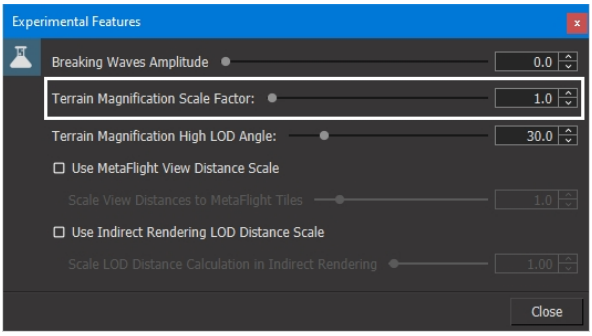
When you zoom the view, all distances used for determining LODs (including terrain) are scaled by the view magnification. In some cases, this is more aggressive than desired for the terrain. To remedy this problem, you can change the amount by which the terrain distance is scaled relative to the view magnification. For example, if view magnification is 8.0 and the Terrain Magnification Scale Factor is 2.0, then the distance used for paging would be scaled by 4.0 (8.0/2.0) rather than by the full factor of 8.0.

Some 3D terrains are not designed to be viewed at higher magnification. Performance may decrease if magnification increases the level of detail of large areas of the terrain. For example, suppose you are looking down a row of trees that is in the center of the view. At normal magnification, there are a certain number of trees at each quality level, say 10 high, 20 medium, and 100 low (for a total of 130 trees in the row). Suppose also that performance will be acceptable only when visualizing 15 or fewer high quality trees. If you magnify the scene 2x, you may now see 20 high, 40 medium, and 70 low quality trees. You see the same number of trees, but performance will decrease because you are visualizing more high quality models.

➤ **To set the scale factor for terrain:**

1. Choose **Settings > Experimental Features**. The Experimental Features dialog box opens (Figure 468).
2. Adjust the Terrain Magnification Scale Factor slider.

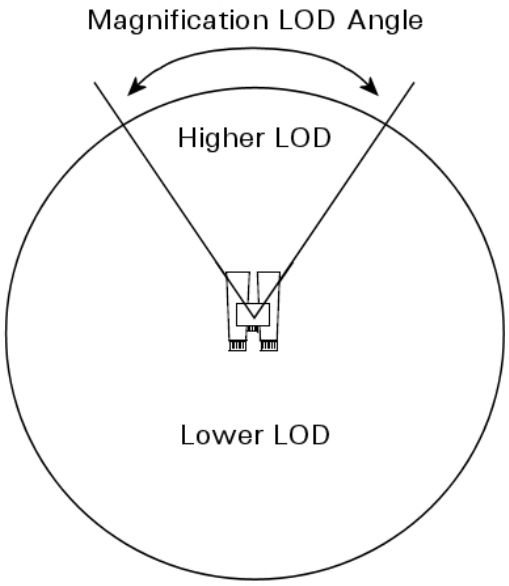
Figure 468. Experimental Features



60.6.4 Setting the Magnification LOD Angle for Terrains

When the observer zooms in, VR-Forces pages in higher LOD tiles for the terrain. This increases memory usage. You can tune performance by restricting high LOD tiles to the areas you are interested in and using lower LOD tiles for the rest of the terrain. The Terrain Magnification High LOD Angle setting lets you adjust the angle within which higher LOD tiles are used (Figure 469).

Figure 469. Terrain magnification LOD angle



If you want the observer to be able to quickly scan a wide angle of terrain, you should increase the magnification LOD angle or you may have to wait for high LOD tiles to page in as you move the observer.

On some terrains, when high LOD tiles are adjacent to low LOD tiles, there can be cracks in the terrain along the boundaries. If you experience this problem, increase the magnification LOD angle.

This feature has a purpose similar to that of LOD centering (described in "60.6.7 Configuring the LOD Center in the Observer Mode" on page 1195). This feature is best used for ground vehicles. We recommend LOD centering for use with aircraft.

This setting is saved when you close VR-Forces.

➤ **To set the magnification LOD angle for terrain:**

1. Choose **Settings > Experimental Features**. The Experimental Features dialog box opens (Figure 468 on the previous page).
2. Adjust the Terrain Magnification High LOD Angle slider.

60.6.5 Zooming In to the Ocean Over Long Distances

When you zoom the observer view to the ocean over very long distances, the ocean might not render at the correct altitude and ships can be hidden (beneath the rendered ocean). The ocean can also interact poorly with the terrain.

If you experience this problem, edit *./data/Environment/Ocean/Triton.config* and set `use-mesh = no`.

If making this change causes performance to decrease, adjust these settings in the file to improve performance:

```
prop-wash-default-fade-time = 60.0
wake-wash-length = 1000
wake-wave-generation-distance = 1.0
wake-kelvin-wave-min-amplitude = 0.1
```

60.6.6 Zooming to Extents

You can zoom the observer to the extents of the terrain or selected simulation objects. When you zoom to extents:

- If no simulation objects are selected and the observer is not attached to a simulation object, the observer view resets to the terrain center. This is the same behavior as pressing **Spacebar**.
- If the observer is attached to a simulation object, the observer resets to the default attached view. This is the same behavior as pressing **Spacebar**.
- If simulation objects are selected and the observer is not attached to a simulation object, the observer view redisplays to show the extents of the selected simulation objects. In the 3D view, this is a top-down view.

Zooming the Observer to the Extents of Selected Simulation Objects

You can quickly change the observer view to show the extents of a selected simulation object or simulation objects. In 2D the observer zooms to the extents. In 3D the observer moves to the correct altitude to view the extents of the simulation objects while looking down at them.

➤ **To zoom the observer to the extents of selected simulation objects:**

1. Select the simulation objects that you want to zoom in on. To zoom to the extents of all simulation objects, do not select any simulation objects.
2. Choose **View > Zoom to Extents**.

Zooming to Simulation Object Extents when a Scenario Loads

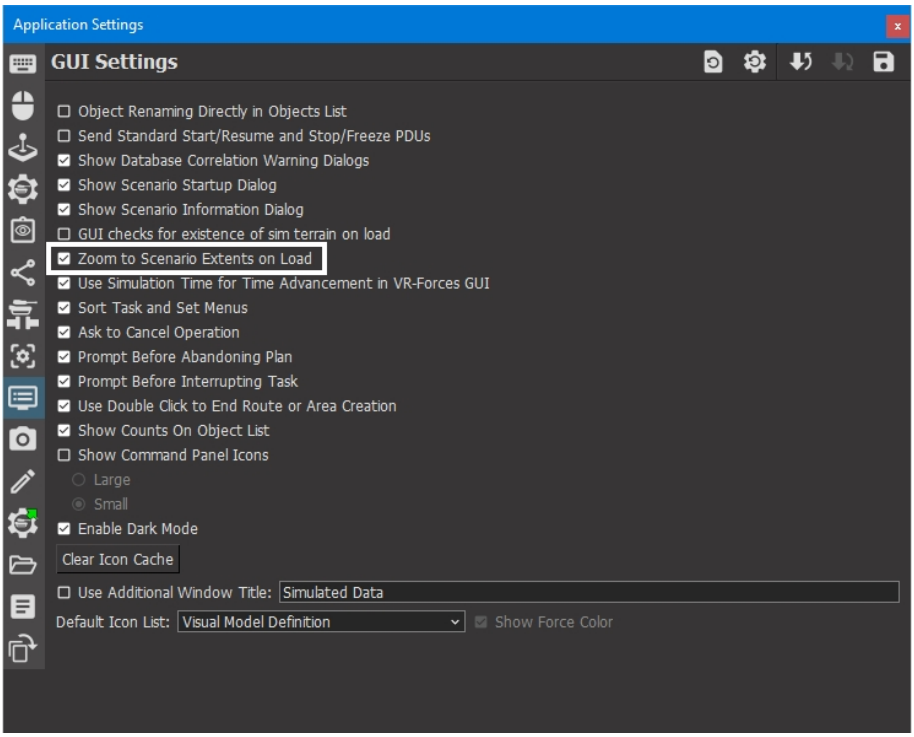
You can configure a scenario so that it zooms to the extents of all simulation objects when the scenario loads. This can be convenient when you have simulation objects located in one part of a large terrain.

> If a scenario has a saved observer view that is designated as the initial view for the scenario, it overrides Zoom to Scenario Extents on Load.

➤ **To zoom to simulation object extents when a scenario loads:**

1. Open the scenario that you want to zoom to extents on load.
2. Choose **Settings > Application**. The Application Settings dialog box opens.
3. Select the GUI Settings page (Figure 470).

Figure 470. GUI Settings page



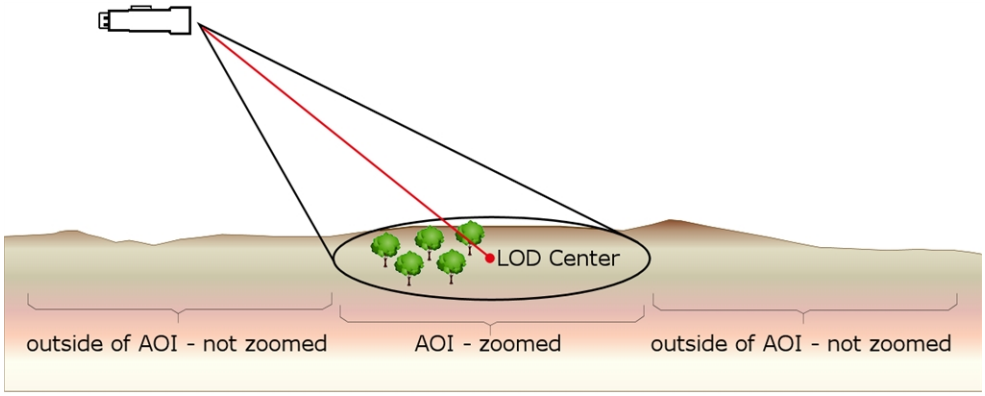
4. Select the Zoom to Scenario Extents on Load check box.
5. Save the scenario. The next time you load the scenario, if the Zoom to Scenario Extents on Load check box is selected, it zooms to the entity extents.

60.6.7 Configuring the LOD Center in the Observer Mode

When the magnification is greater than one, the default behavior is to perform a screen intersection on the area the observer is looking at and improve the LOD at that area and not in other areas. Some areas of the terrain between the observer and the point you have zoomed in on will have a lower LOD, so you might want to change the behavior.

Enabling the LOD center in the observer mode adjusts the LODs for the areas away from the area of interest (AOI) to be lower, which avoids paging or rendering higher LOD models that are not visible in the view. The LOD center uses an intersection with the view vector and the terrain to determine where the AOI is, as shown in (Figure 471).

Figure 471. LOD centering creates an area of interest where the view intersects the terrain



This feature has a purpose similar to that of changing the LOD angle (described in "60.6.4 Setting the Magnification LOD Angle for Terrains" on page 1193). This feature is best used with aircraft. We recommend changing the LOD angle for use with ground vehicles.

➤ **To configure the LOD center in the observer mode:**

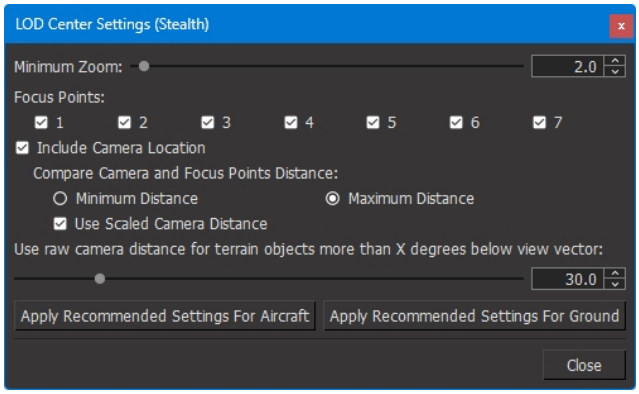
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448 on page 1165).
3. Select or clear the Enable LOD Center check box.
4. Click Close.

Configuring LOD Center Settings

You can configure the observer's LOD center settings to meet your needs. The settings you can adjust are:

- To use the LOD center starting at a specific zoom magnification, adjust the Minimum Zoom slider or enter a value in the box.
- Select the focus point(s).
- To manage the camera settings, select the Include Camera Location check box.
  - Select whether the camera should compare camera and focus points based on the minimum or maximum distance.
  - Select whether to use the scaled camera distance.
  - Specify the number of degrees below the view vector using the slider or enter a value in the box. Note that LOD center must be enabled (zoom out must exceed minimum zoom threshold).

Figure 472. LOD Center Settings (recommended ground settings)



➤ **To configure the LOD center settings:**

1. In the Observer Settings page, select the desired observer mode.
2. Click the LOD Center Settings button. The LOD Center Settings dialog box opens for the specified observer mode.
3. Optionally, click one of the buttons at the bottom to apply the recommended settings for either aircraft or ground.
4. Make further adjustments to the settings as desired.
5. Close the dialog box.

60.7. Centering the Observer on a Simulation Object

You can center the observer on a simulation object from the Object Console Summary Panel, the Last Clicked Object Panel, or the Information dialog box. If you center on a simulation object from the Object Console Summary Panel, the observer is attached to the object. It is not attached using the other methods.

60.7.1 Centering on an Object from the Object Console Summary Panel

If you see that there is a message in the Object Console Summary Panel regarding a simulation object, you may want to quickly view that object. The Object Console Summary Panel lets you center the view on a simulation object. When you center the observer view on a simulation object, the observer attaches to it in Tether mode. For details about Tether mode, see "61.2.7 Tether Mode" on page 1218.

➤ **To center the observer on a simulation object:**

1. In the Object Console Summary Panel, right-click a message from the simulation object you want to center on. A menu displays.
2. Choose **Center Entity**.

60.7.2 Centering on an Object from an Information Panel

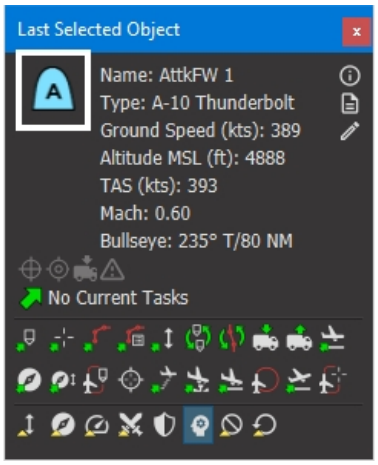
You can center the observer on a simulation object from the Last Clicked Object Panel or the Information dialog box. When you center the observer this way, it is not attached to the object. If the simulation object is moving, it moves away from the center of the display.

If you center on a simulation object in Plan View observer mode, the icon is centered in the window. If you center on a simulation object in Stealth observer mode, the observer stays in the current orientation relative to the centered object. If you press the **Spacebar**, the view changes to a top down view, similar to Plan View mode.

> If the observer is attached to a simulation object, you cannot center the observer on a different object using this procedure.

➤ To center the observer on a simulation object from the Last Clicked Object Panel or the Information dialog box, click the simulation object icon on the panel (for example, Figure 473).

Figure 473. Simulation object icon



60.8. Simulating Camera Jitter

You can simulate the jitter that cameras attached to aircraft or unsteady human hands may experience.

➤ **To simulate camera jitter:**

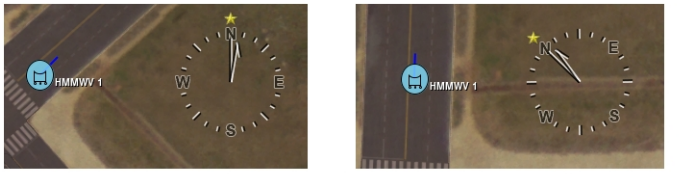
1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448 on page 1165).
3. In the Camera Shake section, select Allow Camera Shake.
4. Optionally, adjust the Intensity slider or change the value in the box.

60.9. Forcing Orientation North

In the 2D view, by default the observer's heading is forced to stay north (up), regardless of the heading of the simulation object to which it is attached. You can disable this setting, in which case if you attach to a simulation object, the view of the terrain rotates to match the heading of the attached simulation object.

Figure 474 illustrates the view when the observer is attached to an entity with orientation forced to north or matched to that of the entity.

Figure 474. Forcing orientation north



➤ **To enable or disable a northward orientation:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 464 on page 1188).
3. In the Observer Mode list, select Plan View.
4. In the list of observer settings, select or clear Orient North.
5. Optionally, you can define whether north is true or magnetic. See "60.9.1 Specifying True, Magnetic, or Grid North" below.

You can also enable or disable northward orientation by choosing **Observer > Orient North**.

60.9.1 Specifying True, Magnetic, or Grid North

When the observer is set to Orient North, you can specify which north to use for the plan view (2D). True north, also known as geodetic north, is the direction pointing to the geographic location of the North Pole. Magnetic north, on the other hand, is the point where the magnetic field points down. Lastly, grid north is the direction indicated by the vertical grid lines on ordnance survey maps.[1]

➤ **To specify the type of north:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448 on page 1165).

[1]In VR-Forces, magnetic north is determined by the World Magnetic Model (WMM). (The dates in the simulation are the dates used for the magnetic model information.) To learn more about the WMM, visit https://www.ngdc.noaa.gov/geomag/WMM.

3. In the Observer Mode list, select Plan View.
4. In the Map Alignment Preference list, select True North, Magnetic North, or Grid North.

60.10. Enabling and Disabling View Constraints (3D Only)

View constraints control how the observer behaves when it approaches solid objects in the view, such as the ground, buildings, and vegetation. When constraints are enabled, the observer has a 1 meter bounding sphere that prevents it from moving through terrain or props.

> If view constraints are enabled, you may not be able to move the observer into buildings to view and place entities. Changing the near clipping plane or transparency will let you see through walls, but in constrained mode, you will still not be able to move the observer through the walls.

➤ **To enable or disable view constraints, use one of the following methods:**

- Press **/** (forward slash).
- Choose **Observer > Constrain to Terrain**.
- On the Observer Control Panel (Figure 465 on page 1189), select or clear the Terrain Constrained check box.

60.11. Controlling Navigation Speed

When you navigate through the terrain, there are times you want to move quickly, covering large amounts of terrain, and times you want to move slowly, such as when you are moving down a street or within a building. VR-Forces has options for controlling navigation speed:

- Set the navigation speed manually. The observer always moves at the specified speed.
- Use speed scaling to automatically move at different speeds under different circumstances. You can change the rate at which speed is scaled.

In either case, the controls for changing speed are the same. Their effect varies depending on whether speed scaling is enabled.

60.11.1 Enabling or Disabling Speed Scaling

When speed scaling is enabled, VR-Forces makes the following assumptions:

- If the observer is unattached, the higher above the terrain it is, the faster you want to move.
- The closer you are to an attached object, the more slowly you want to move.

➤ **To enable or disable speed scaling, use any of the following methods:**

- On the Observer Control Panel (Figure 465 on page 1189), select or clear the Scaled Speed check box.
- Choose **Observer > Speed Scaling**.
- Press the multiply key (*****) on the keypad (not the asterisk key).

Setting the Speed Scaling Mode

When speed scaling is enabled, VR-Forces scales observer speed based on the its height above the terrain. Even if the observer is higher than all terrain points, speed can still be affected if there are changes to the terrain, such as mountains. This can be annoying if it happens unexpectedly. Therefore, you can choose to base speed scaling on height above ground level, or height above mean sea level. In that latter case, changes to the terrain would not affect the speed at which the observer moves. If you set speed scaling to use height above mean sea level, you can specify an offset.

➤ **To set the speed scaling mode, use one of the following methods:**

- On the Observer Control Panel (Figure 465 on page 1189), set the Speed Scaling Mode to Above Ground Level (AGL) or Above Mean Sea Level (MSL)
- On the Display Settings dialog box, Observer Settings page, set the Speed Scaling Mode to Above Ground Level (AGL) or Above Mean Sea Level (MSL)

60.11.2 Changing the Observer's Movement Speed

You can change the observer's linear, rotational, and orbital speed. If speed scaling is enabled, these controls change the speed gain. If speed scaling is disabled, the controls change the absolute speed.

➤ **To change the observer's speed, use any of the following methods:**

- Adjust one of the Observer Speed sliders on the Observer Control Panel (Figure 465 on page 1189) or the Observer Settings page in the Display Settings (Figure 448 on page 1165). The default (center of the slider) is 1. The range is 0.1 to 10.
- Press one of the following keys to change linear speed:
  - **Keypad 7** (faster) or **1** (slower).
  - **Ctrl**+mouse wheel forward or back.
  - Movement key + mouse wheel forward or back.
- Press **Keypad +** (faster) or **-** (slower) to change orbital speed.

When you change the observer's speed using the keyboard or mouse, if the Observer Control Panel or the Observer Settings page is open, you can see the scale slider move to the new setting.

60.12. Saving and Recalling Views

At any given moment, the state of the observer is called its view. You can capture views so that you can quickly return to them. You can also save them for future use. VR-Forces automatically saves observer views as part of the scenario (*scenario_name.osrx*). You can also save views independently of scenarios.

VR-Forces supports the following use cases for saving observer views:

- Associate views with a terrain. Save one or more absolute observer mode views and associate them with a terrain. The views are loaded automatically when the terrain loads. You can specify a default view that the observer jumps to immediately.
- Save a mix of absolute mode and attached mode views to a file. This use case is best when you want to create a set of views for use with multiple scenarios. You can import the file into each scenario that you want to use it with.

> Nothing prevents you from including observer views that are attached to a simulation object in a terrain's saved views file, but they will not work unless the specific simulation object is available in the simulation.

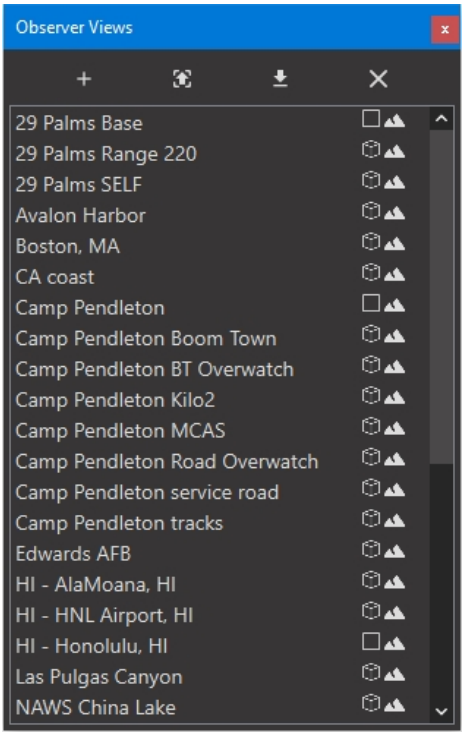
If the observer is not attached to a simulation object, its view is its position and orientation in the simulated environment. If the observer is attached to a simulation object, a view saves the:

- Attach mode.
- Positional offset.
- Marking text.

> When you save a view in the 2D view, the X and Y values are saved. The zoom level is not saved.

Saved views are managed in the Observer Views Panel (Figure 475).

Figure 475. Observer Views Panel



The buttons in the Observer Views Panel:

- Add the current view as a new observer view.
- Save the views to a file.
- Load a saved views file and append it to the list.
- Clear the list.

The icons in the Observer Views list are:

- 3D view, such as Stealth, XR, or Out-the-Window.
- 2D view, such as Plan View.
- View is associated with the terrain.
- Default view for the terrain.
- Default scenario view.

Saved views are keyed to the marking text of the object to which the observer is attached.

➤ **To save the current view:**

1. Choose **Views > Observer Views Panel**. The Observer Views Panel opens (Figure 475).
2. Click the Add button (+). The current view is added to the list.

60.12.1 Recalling a View

➤ To recall a view, select it in the Observer Views Panel. The view changes immediately.

Animating Transitions Between Observer Views

Normally, when you select an observer view, the observer jumps immediately to the new view. However, VR-Forces can smoothly transition from the current observer view to a saved view.

➤ To transition between observer views, hold down the **Ctrl** key and click the view in the Observer Views Panel that you want the observer to move to.

ⓘ If the observer view that you select uses a different observer mode (for example, you would be moving from Plan View mode to Stealth), VR-Forces does not change the observer mode. The view moves to an approximation of the desired view in the current observer mode.

60.12.2 Deleting Views

➤ **To delete a view:**

1. In the Observer Views Panel, select the view you want to delete.
2. Right-click on the view and then choose **Delete Observer View** from the context menu.

Clearing the View List

When you clear the list, VR-Forces removes all scenario-specific views and marks views that are associated with the terrain as hidden. When you load a scenario that has views associated with the terrain, VR-Forces ignores any views that are marked as hidden in the scenario. If additional views were associated with the terrain after the observer views list was cleared for the scenario, they are not marked as hidden and, therefore, are loaded. (Hidden views are recorded in the scenarios *.scn* file.)

➤ To remove all views from the list, in the Observer Views Panel, click the Clear List button (✕).

60.12.3 Loading Views Automatically When You Load a Terrain

You can associate a saved views file with a terrain so that it loads automatically when the terrain loads. You can also specify that one of the views be the default view. This is most useful when you create a new scenario. Once you save a scenario, the observer views saved with the scenario take precedence over the views associated with the terrain.

You associate views with a terrain by saving the views to a file in the same directory as the MTF file (by default, *./SharedData/19/latest/TerrainData/TerrainConfiguration*). The file must have the same name as the terrain and the extension *.osrx*. The next time you open the terrain, the views are listed in the Observer Views Panel. An icon (⛰) next to each view shows that it is associated.

The views associated with the terrain are not saved as part of the scenario. Each time you load a scenario, if there are views associated with the terrain, they are loaded (unless they are marked as hidden in the scenario file). Then the views saved in the scenario are loaded.

> The file used for associating views with a terrain is different from the file that is created when you manually save views to a file, as described in "60.12.5 Saving Views to a File" on the next page.

➤ **To associate views with a terrain:**

1. Click the Export Views button (). A file browser dialog box opens.
2. Change directory to the directory where the terrain's MTF file is, usually *./SharedData/19/latest/TerrainData/TerrainConfiguration*.
3. Save the views to a file that has the same name as the MTF file.

Specifying a Default Observer View for a Terrain

If you associate views with a terrain, you can specify that one of the views be the default view to use. A star is added to the association icon for default views ().

You can specify a default 3D view and a default 2D view. When you load a terrain, if there are two default views and the current observer mode is a 3D mode, it uses the 3D default view. If the current observer mode is a 2D mode, it uses the 2D default view.

➤ **To specify a default observer view:**

1. Right-click the observer view that you want to be the default.
2. On the shortcut menu, choose **Set As Default Terrain View**.

Unsetting a Default View for a Terrain

➤ **To unset a default observer view:**

1. Right-click the default observer view.
2. On the shortcut menu, choose **Unset Default Terrain View**.

60.12.4 Specifying the Scenario Startup View

If a scenario has observer views, you can specify that one of them be the view selected when the scenario loads. (The observer view is not selected when you rewind a scenario.) If a scenario has a default observer view and the terrain has a default observer view, the scenario observer view is used.

➤ **To specify a view as the startup view:**

1. Right-click the view.
2. On the shortcut menu, choose **Set Default Scenario View**. A star icon next to the view shows that it is the default view.

Unsetting a Default Scenario View

➤ **To unset a default observer view:**

1. Right-click the default observer view.
2. On the shortcut menu, choose **Unset Default Scenario View**.

60.12.5 Saving Views to a File

You can save views to a file independently of the scenario and load them at a later time. The default location for saved views is *./appData/settings/vrfGui*. When you create a 2D view for a terrain that includes 2D View Layers, that view includes the currently specified layer.

Saving views to a file is not the same thing as associating views with a terrain. When you associate views with a terrain, they load automatically and you can specify a default view to use immediately. When you save views to a file, you need to load them manually, or from the command line. You cannot specify a default view.

Associating views with a terrain is most beneficial for saving a set of Absolute mode observer views that you always want to be available to the terrain, regardless of the simulation that you are connected to. Saving views to a file manually is most beneficial for saving views when the observer is attached to simulation objects in a simulation or for simulation-specific Absolute mode views.

➤ **To save all views to a file:**

1. In the Observer Views Panel, click the Export Views button (). The Save Observer Views dialog box opens.
2. Select the file to which you want to save the views, or type the name of a new file.
3. Click OK.

Saving an Individual View to a File

You can save a single view to a file.

➤ **To save an individual view to a file:**

1. Right-click the view that you want to save.
2. On the menu, choose **Save Observer View**. The Save Observer View File dialog box opens.
3. Type a name for the file.
4. Click Save.

60.12.6 Loading Views

You can load saved views and append them to the list in the Observer Views Panel.

If a scenario has observer views saved as part of the scenario, they are loaded automatically when you load the scenario. If you have saved views independently of the scenario, you can load them and append them to the list in the Observer Views Panel.

To load a view, you can also drag and drop the *.osrx* file onto the Observer Views Panel and then select it.

➤ **To load views and append them to the list of views:**

1. In the Observer Views Panel (Figure 475 on page 1203), click the Import and Append Views button (). The Open Observer Views dialog box opens.
2. Select the saved views file that you want to load.
3. Click OK.

60.13. Using the Go To Location Panel

The Go To Location panel allows you to move the current observer to the specified address, latitude and longitude, or MGRS coordinates.

➤ **To go to a location:**

1. Open the Go To Location panel (**View > Go To Location**).
2. Select the method you want to use and then enter the appropriate information:
   - To go to an address, select Go To Address and then enter the complete address. This requires an internet connection. If VR-Forces cannot find some address to match, it displays an alert icon. Hover the mouse over it for more information.

     You can also enter the code for an airport and use that to go to the airport's location.
   - To specify the location's coordinates, select a coordinate system from the Go To Location drop-down list and then enter the coordinates in the text boxes.
3. Click Go. The Go button becomes active when you have entered the required information.

   If you entered an address but the current observer did not jump to the address you specified, there was an error. An icon displays next to the address field. Hover the mouse over the icon to read the error message.

60.14. Controlling the Observer from Other Applications

The VR-Forces GUI can respond to view control messages sent by applications such as the MAK Data Logger. It also responds to view control messages sent from entity plans and global plans. You can write VR-Link-based applications to send view control messages to VR-Forces using the classes in the VR-Vantage Control Toolkit.

You can use view control messages to:

- Specify the attach mode and simulation object.
- Set the observer's position and orientation.

Acceptance of view control messages is a global setting.

> The view control PDUs are experimental, and are not an official part of DIS. They are of PDU kind 202.

60.14.1 Enabling the Processing of View Control Messages

➤ To enable or disable the processing of view control messages, choose **Settings > View Controls**.



61. Attaching the Observer to Objects

This chapter describes attachment modes and how to attach the observer to simulation objects and props.

61.1. Attaching the Observer to a Simulation Object or Prop 1210
- 61.1.1 Attaching the Observer to Another Observer 1212
- 61.1.2 Attaching to an Articulated Part 1212
- 61.1.3 Attaching the Observer in Mimic Track or Tether Track Mode 1212
- 61.1.4 Attaching the Observer in Mimic Location Track or Tether Location Track Mode 1213
- 61.1.5 Filtering the Object Attachment List 1213
- 61.1.6 Detaching from a Simulation Object 1213
- 61.1.7 Hiding the Attached Model (3D View Only) 1213
- 61.1.8 Attaching to Props 1214
- 61.1.9 Improving Performance when Attaching to Ground Objects 1214

61.2. Attach Modes in the 3D View 1214
- 61.2.1 Absolute Mode 1215
- 61.2.2 Follow Mode 1215
- 61.2.3 Mimic Mode 1215
- 61.2.4 Mimic Track Mode 1216
- 61.2.5 Mimic Location Track Mode 1217
- 61.2.6 Space Follow Mode 1217
- 61.2.7 Tether Mode 1218
- 61.2.8 Tether Track Mode 1219
- 61.2.9 Tether Location Track Mode 1219
- 61.2.10 Track Mode 1219

61.3. Attach Modes in the 2D View 1220

61.4. Attaching the Observer at Startup 1221

61.1. Attaching the Observer to a Simulation Object or Prop

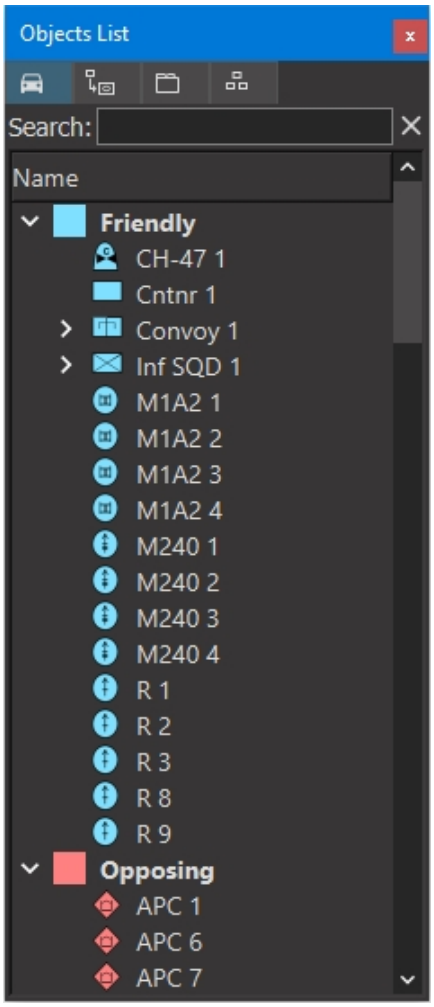
You can attach the observer to a simulation object and follow it as it moves through the simulation. You can also attach the observer to props and to other observers.

> This procedure applies to Follow, Tether, Track, Mimic, and Space Follow modes. To attach in Tether Track or Mimic Track mode, see "61.1.3 Attaching the Observer in Mimic Track or Tether Track Mode" on page 1212. To attach in Tether Location Track or Mimic Location Track mode, see "61.1.4 Attaching the Observer in Mimic Location Track or Tether Location Track Mode" on page 1213.

➤ **To attach the observer to a simulation object or prop, do one of the following:**

- Right-click an object in the Objects List Panel (Figure 476) or on the terrain and select **Observer Attach > Attach** *mode*, where *mode* is one of the supported attach modes. (For details about attach modes, see "61.2. Attach Modes in the 3D View" on page 1214 and "61.3. Attach Modes in the 2D View" on page 1220.)

Figure 476. Objects List Panel



- Select a simulation object or prop. (For details about selecting simulation objects and props, see and "Selecting Props" in the *Adding Content to MAK ONE Applications Reference Manual*.) On the Attachments Panel (Figure 477), or on the Attach Object Toolbar (Figure 478), click the Attach button (🔗) to attach with the current attachment mode.

Figure 477. Attachments Panel

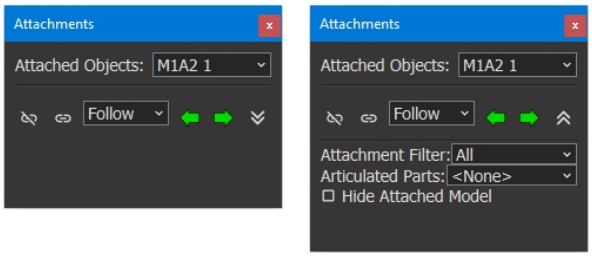


Figure 478. Attach Object Toolbar



- On the Attachments Panel or Attach Object Toolbar, click the next or previous arrows to attach to the next (or previous) object in the object list.
- **Shift**+click the object you want to attach to.
- Press greater than (**>**) to attach to the next simulation object or prop in the Objects List without regard to the attachment filter. Press less than (**<**) to attach to the previous object.
- Press period (**.**) to attach to the next simulation object or prop allowed by the attachment filter. Press comma (**,**) to attach to the previous object.

- When you select the next or previous object, VR-Forces chooses next and previous based on the order in which the objects were discovered. This might represent the order in which they are listed in the Objects List Panel.
- When you move the attachment from one object to the next, VR-Forces preserves whatever changes you have made to the observer's view relative to the attached object. For example, if you moved the observer so that it is not looking at the attached object, when you attach to the next simulation object, the observer will not be looking at this object. It will be in the same relative position to the object that you previously established. To quickly return to the default offset for the attachment mode, press **Spacebar**.

61.1.1 Attaching the Observer to Another Observer

You can attach observers to other observers if you have more than one window open.

➤ **To attach an observer to another observer:**

1. Filter the Objects List to show All objects or to show observers.
2. Select the window whose observer you want to attach to another observer.
3. In the Objects List, select the observer you want to attach to.
4. Right-click and select **Observer Attach > Attach** *mode*, where *mode* is one of the supported attach modes.

61.1.2 Attaching to an Articulated Part

You can attach the observer to an articulated part on an entity, such as a turret. When you attach to an articulated part, the observer moves with the articulated part instead of the main body of the entity.

➤ **To attach to an articulated part:**

1. Attach the observer to an entity, as described in "61.1. Attaching the Observer to a Simulation Object or Prop" on page 1210.
2. On the Attachments Panel (Figure 477 on the previous page), click the down arrow. The Attachments Panel expands.
3. In the Articulated Parts list, select the part to which you want to attach the observer.

61.1.3 Attaching the Observer in Mimic Track or Tether Track Mode

To attach the observer in Tether Track and Mimic Track modes, you must specify a primary simulation object and a secondary simulation object (the tracked simulation object).

➤ **To attach the observer in Mimic Track mode or Tether Track mode:**

1. Right-click the primary simulation object in the Objects List Panel or in the window. A shortcut menu displays.
2. Choose **Observer Attach > Attach Mimic Track** (or **Attach Tether Track**). The cursor changes to show an arrow and a Plus sign.
3. Click the secondary simulation object in the Objects List or in the window.

61.1.4 Attaching the Observer in Mimic Location Track or Tether Location Track Mode

To attach the observer in Tether Location Track and Mimic Location Track modes, you must specify a primary simulation object and a location.

➤ **To attach the observer in Mimic Location Track mode or Tether Location Track mode:**

1. Right-click the primary simulation object in the Objects List Panel or in the window. A shortcut menu displays.
2. Choose **Observer Attach > Attach Mimic Location Track** (or **Attach Tether Location Track**). The cursor changes to show an arrow and a Plus sign.
3. Click a location in the display window.

61.1.5 Filtering the Object Attachment List

Although you can attach the observer to simulation objects and props, it is likely that at any given time you will only want to attach to one type of object (usually simulation objects). To reduce the number of objects that you have to iterate through when using Attachments Panel arrows to pick the next or previous object, you can filter the attachment list.

ⓘ The filter for buoys and beacons applies to S-57 feature data, not buoys that are simulation objects, such as sonobuoys.

➤ **To filter the attachment list:**

1. On the Attachments Panel (Figure 477 on page 1211), click the down arrow. The Attachments Panel expands.
2. In the Attachment Filter list, select the type of scene element that you want to attach to.

61.1.6 Detaching from a Simulation Object

➤ To detach the observer from the simulation object to which it is attached, click the Detach button (🔗) on the Attachments Panel (Figure 477 on page 1211), or press the **Z** key.

61.1.7 Hiding the Attached Model (3D View Only)

You can hide the model for the simulation object to which the observer is attached.

➤ **To hide the attached model:**

1. On the Attachments Panel (Figure 477 on page 1211), click the down arrow. The Attachments Panel expands.
2. Select the Hide Attached Model check box.

61.1.8 Attaching to Props

Since props do not move, attaching the observer to a prop does not result in any observer movement. The principal benefit of attaching to a prop is to let you quickly refocus the observer on a particular location in the scene. If you attach in Track mode, using the keyboard movement keys allows you to orbit around the prop.

61.1.9 Improving Performance when Attaching to Ground Objects

When the observer is attached to air entities, Z-fighting can affect visual quality. To prevent this problem, VR-Forces needs to calculate the entity's height above the terrain. However, the calculation can affect performance. Unfortunately, when this calculation is enabled (the default), it must be applied to all entity types. If you know that you will want to attach the observer to ground simulation objects, but not air simulation objects (or if you do not care about Z-fighting), you can disable the height above terrain calculation, which may improve performance when the observer is attached to simulation objects. To disable the height above terrain calculation, start VR-Forces with the `--suppressHatIntersectOnAttach | -S` command-line option.

61.2. Attach Modes in the 3D View

Attach modes are combinations of position and orientation models and constraints that produce a useful behavior for the observer. Except for Absolute mode, attach modes describe how movement of the simulation object to which the observer is attached affects the position and orientation of the observer.

VR-Forces supports these attach modes:

- Absolute: The observer is not attached to a simulation object; it moves in space independently of all simulation objects.
- Follow: The observer position and heading changes with the simulation object's position and heading.
- Mimic: The observer position and orientation changes with the simulation object's position and orientation.
- Mimic Track: The observer's position changes with the simulation object's position while its orientation is focused on a second entity.
- Mimic Location Track: The observer's position changes with the simulation object's position while its orientation is focused on a location.
- Space Follow: The observer is a fixed distance from the simulation object and the simulation object model is scaled to be observable from a great distance.
- Tether: The observer's position changes with the simulation object's position.
- Tether Track: The observer's position changes with the simulation object's position while its heading is focused on a second simulation object.
- Tether Location Track: The observer's position changes with the simulation object's position while its heading is focused on a location.

- Track: The observer's heading changes to track the simulation object's position and heading.

The attach modes are described in detail in the following sections. For details about how to attach the observer to a simulation object, see "61.1. Attaching the Observer to a Simulation Object or Prop" on page 1210.

61.2.1 Absolute Mode

In Absolute, or free-fly mode, the observer is not attached to a simulation object. Movement of simulation objects does not affect the observer in any way.

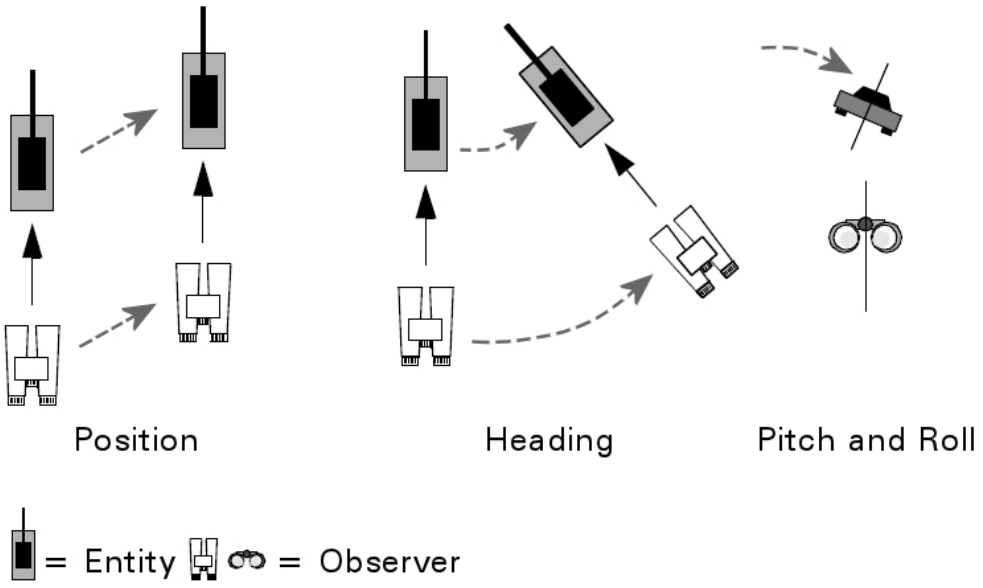
61.2.2 Follow Mode

In Follow mode, the observer maintains its relationship to the simulation object:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading and position change to maintain the offset.
- When the simulation object's pitch or roll change, the observer pitch and roll do not change.

Figure 479 illustrates the characteristics of Follow mode.

Figure 479. Follow mode



61.2.3 Mimic Mode

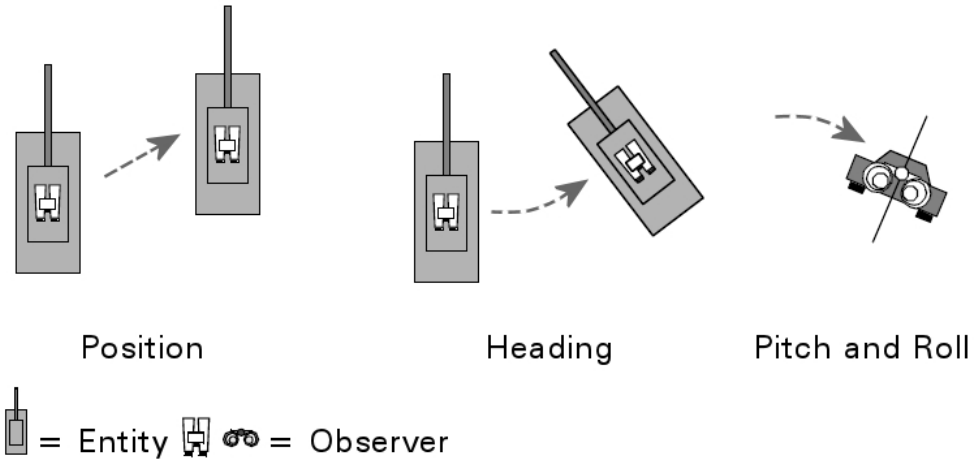
In Mimic mode, the observer behaves as though it is rigidly connected to the entity. Use Mimic mode to create the effect of riding in a vehicle cockpit, or of following a vehicle from an external location while connected to it by a solid, invisible rod. The observer maintains its relationship to the simulation object:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading and position change to maintain the offset.
- When the simulation object's pitch or roll change, the observer's pitch and roll change and the observer's position changes to maintain its offset.

When cockpit displays are enabled, you cannot change the position or orientation of the observer relative to the entity.

Figure 480 illustrates the characteristics of Mimic mode.

Figure 480. Mimic mode



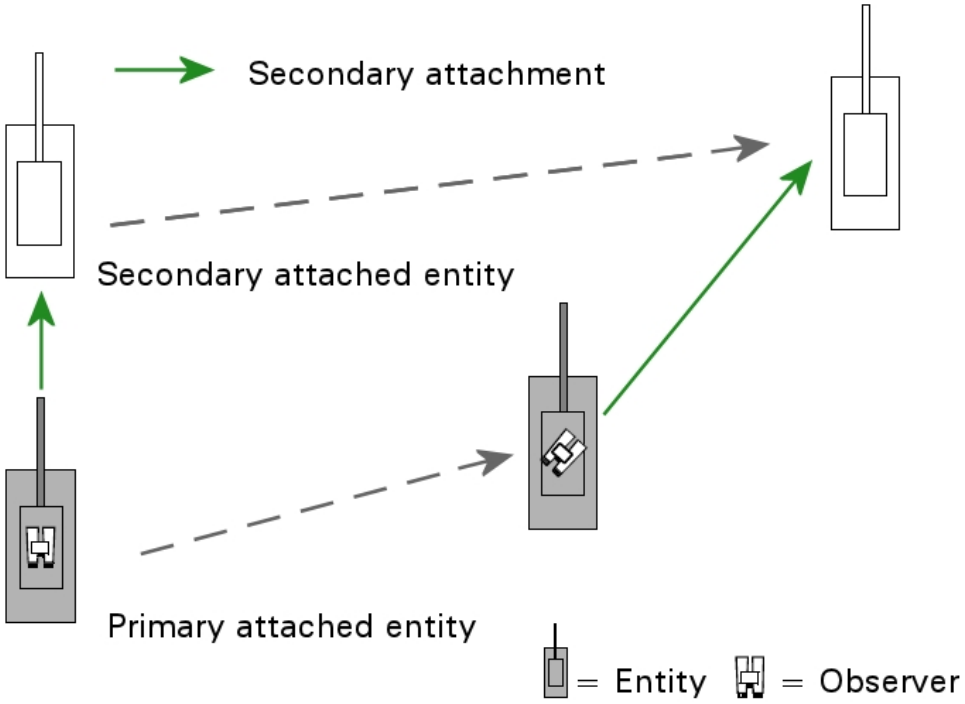
61.2.4 Mimic Track Mode

Mimic Track mode combines Mimic mode and Track mode. The observer's position changes with that of the entity to which it is attached, as in Mimic mode, while its orientation changes to remain focused on a secondary, tracked entity.

Orientation controls are disabled.

Figure 481 illustrates the characteristics of Mimic Track mode.

Figure 481. Mimic Track mode



61.2.5 Mimic Location Track Mode

Mimic Location Track mode works like Mimic Track mode except it tracks a location on the ground rather than an entity.

61.2.6 Space Follow Mode

Space Follow mode attaches the observer above an entity, looking down at it. Although it can be used with any entity, its primary purpose is for attaching to space objects such as satellites. You can also usefully use it with fixed-wing and rotary-wing entities.

The observer is attached at a fixed distance, determined by the near clipping plane for the object. For a satellite, this distance is approximately 500,000 meters. Since it is not possible to see regular models at that distance, the attached object is scaled to be visible. (You cannot change the scaling of the attached object.) Other objects are not scaled (except to the extent that model scaling might be enabled).

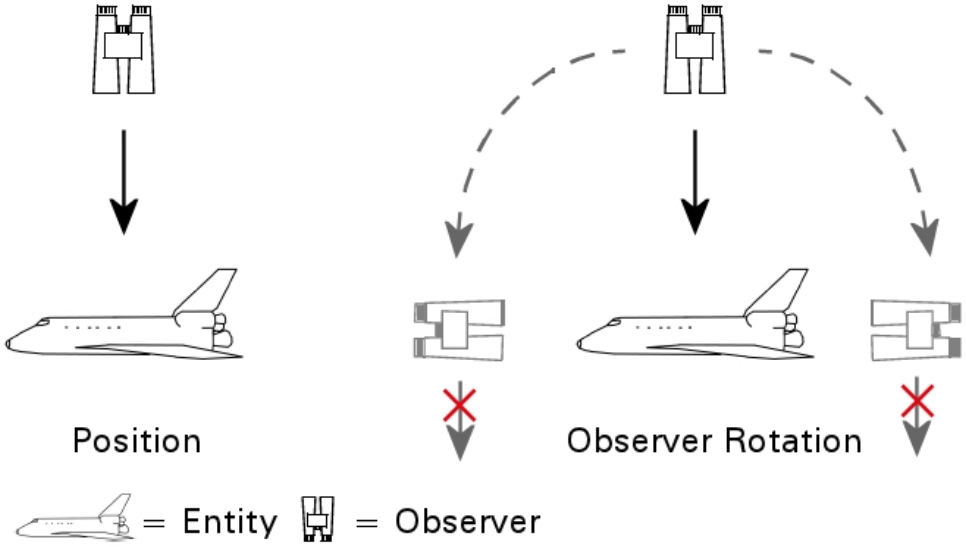
The observer maintains its relationship to the simulation object:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading changes.
- When the simulation object's pitch or roll changes, the observer's pitch or roll does not change.

You cannot move the observer towards or away from the simulation object. You can rotate the observer around the entity, but you cannot go below the horizontal plane of the entity.

Figure 482 illustrates the characteristics of Space Follow mode.

Figure 482. Space Follow mode



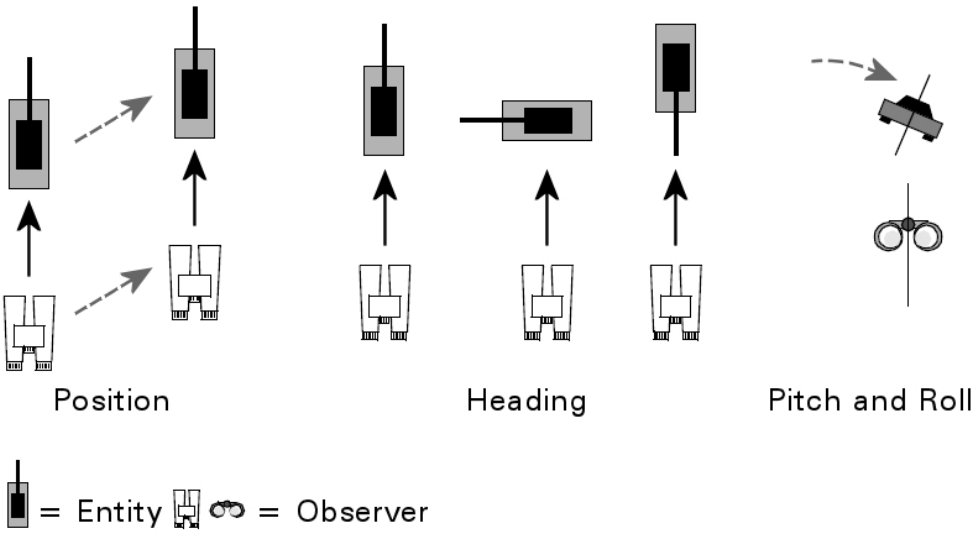
61.2.7 Tether Mode

In Tether mode, the observer's maintains a constant offset from the simulation object, but it is not affected by the simulation object's heading, pitch, or roll:

- When the simulation object's position changes, the observer's position changes to maintain its offset.
- When the simulation object's heading changes, the observer's heading does not change.
- When the simulation object's pitch or roll change, the observer's pitch and roll do not change.

Figure 483 illustrates the characteristics of Tether mode.

Figure 483. Tether mode



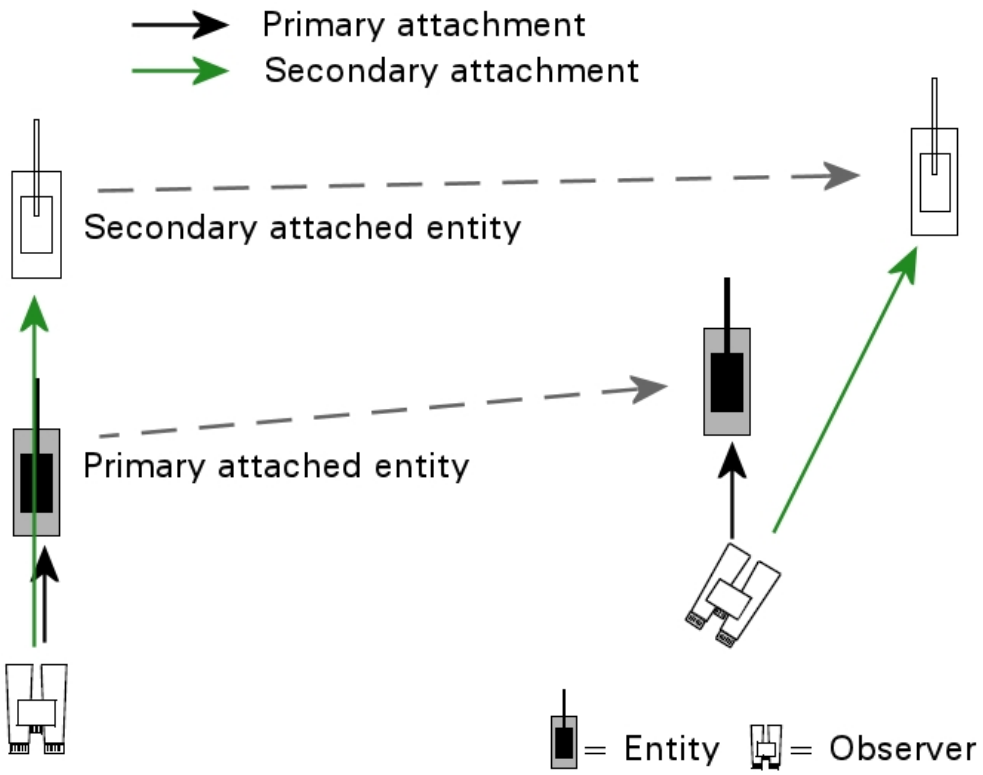
61.2.8 Tether Track Mode

Tether Track mode combines Tether mode and Track mode. The observer's position changes with that of the entity to which it is attached, as in Tether mode, while its orientation changes to remain focused on a secondary, tracked entity.

Orientation controls are disabled.

Figure 484 illustrates the characteristics of Tether Track mode.

Figure 484. Tether Track mode



61.2.9 Tether Location Track Mode

Tether Location Track mode works like Tether Track mode, except that it tracks a location on the ground instead of an entity.

61.2.10 Track Mode

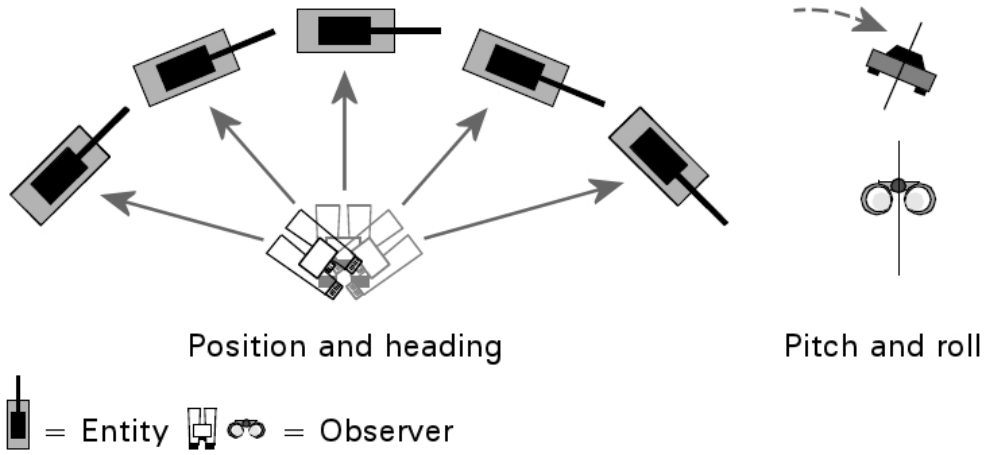
In Track mode, the observer's heading changes to track the simulation object's movement, keeping it centered in the field of view. The observer's position does not change. The observer maintains its relationship to the simulation object:

- When the simulation object's position changes, the observer's heading changes to keep the entity centered in the observer's view; the observer's position does not change.

- When the simulation object's heading changes, the observer's heading does not change.
- When the simulation object's pitch or roll change, the observer's pitch and roll do not change.

Figure 485 illustrates the characteristics of Track mode.

Figure 485. Track mode



61.3. Attach Modes in the 2D View

The 2D view supports Follow mode and Tether mode. In Follow mode, the heading of the observer changes with the entity (if Orient North is disabled); in Tether mode, it does not. You can use the standard movement keys (**W**, **A**, **S**, **D**) to offset the observer with respect to the simulation object's position. You can use the orbit keys (left arrow, right arrow) to offset the observer's orientation with respect to the entity's orientation (in Follow mode) or the terrain.

If the observer is attached to an entity in the 3D view and you switch to the 2D view, the attach mode maps to either Follow or Tether modes, as shown in Table 51. If you then switch back to the 3D view, the original attach mode is restored.

Table 51.
3D to 2D attachment mode mapping

3D Attach Mode	2D Attach Mode
Follow	Follow
Mimic	Follow
Mimic Track	Follow
Mimic Location Track	Follow
Space Follow	Follow
Tether	Tether

Table 51.
3D to 2D attachment mode mapping (continued)

3D Attach Mode	2D Attach Mode
Tether Track	Tether
Tether Location Track	Tether
Track	Tether

61.4. Attaching the Observer at Startup

To attach the observer to an entity when you load a scenario, save a view and specify it as a favorite, as described in "60.12.4 Specifying the Scenario Startup View" on page 1205.



62. Inset Views, Sensor FOVs, HUDs, and Depth of Field

This chapter describes several kinds of special secondary windows plus head up displays.

62.1. Inset Views 1224
62.2. Displaying Cockpit Displays 1225
62.3. Displaying Instrument Panels 1226
62.3.1 Configuring Entities to Use Instrument Panels 1228
62.4. Displaying Sensor Views 1228
62.4.1 Displaying a Sensor Field of View Cone 1229
62.4.2 Controlling the Sensor View 1230
62.5. Using the Depth of Field Effect 1233
62.5.1 Configuring Depth of Field 1235

62.1. Inset Views

You can display inset views of individual simulation objects. Figure 486 shows a helicopter hovering over the Ala Moana area in the *MAK Earth (online).mtf* terrain with an inset view of cars driving through the town.

Figure 486. Inset view



⚠ By default, when you create an inset view, the observer in the inset view is attached to the simulation object in Mimic mode. This has the side effect of causing the 3D model in the main window to disappear. (The reverse occurs if you attach the observer in the main window to the simulation object in Mimic mode.) To make the model reappear, select the inset view and change the attach mode or detach the observer.

You can change the observer position and orientation in an inset view using keyboard controls. You can also edit window and channel properties for the window in the Display Engine Configuration Editor. (For information about the Display Engine Configuration Editor, see "83. Display Engine and Window Management" on page 1597.)

When you create an inset view, the title bar contains the simulation object's name. However, the window is not actually tied to the simulation object and you can change the view in the inset view and view other simulation objects. The title bar will continue to display the name of the simulation object for which the window was created.

➤ **To open an inset view:**

1. Select the object for which you want to create an inset view.
2. Choose **Objects > Create Window > Create Inset View**.

You can also right-click on the object and then choose **Create Window > Create Inset View**.

62.2. Displaying Cockpit Displays

VR-Forces can display cockpit displays created with GL Studio. It includes cockpits for fixed-wing, rotary-wing, and ground vehicle entities. As with other entity effects, cockpits have schemas and model definitions that you map to entity types. For details about cockpit display models, see "Adding New Cockpit Display Models" in the *Adding Content to MAK ONE Applications Reference Manual*.

VR-Forces supports 2D cockpits (Figure 487) and 3D cockpits (Figure 488). A 2D cockpit is fixed to the center of the screen and you cannot zoom in or out on the instrument panel. A 3D cockpit is aligned with the front of the vehicle. If you move the eyepoint, the instrument panel moves in or out of view as you would expect. You can zoom in and out on the instruments.

Figure 487. 2D cockpit



Figure 488. 3D cockpit

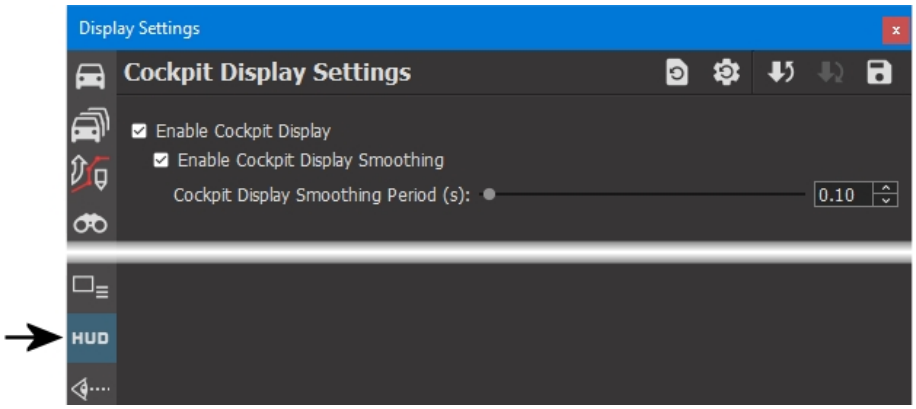


- Cockpits display only when the observer is attached to an entity in Mimic mode.
- VR-Forces includes one 3D cockpit, for the F-16 fighter jet.
- Cockpits are not supported on Linux.

➤ **To display cockpits:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Cockpit Display Settings page (Figure 489).

Figure 489. Cockpit display settings



3. Select or clear the Enable Cockpit Display check box.
4. Optionally, select the Enable Cockpit Display Smoothing check box. This smooths the display in the dynamic widgets, such as orientation or velocity, so they show a smoothed value rather than instantaneous changes.
5. Attach the observer to an entity in Mimic mode.

You can also enable and disable cockpit displays by choosing **Settings > Cockpit Displays**.

62.3. Displaying Instrument Panels

Instrument panels are inset windows that display a vehicle instrument such as an altimeter or artificial horizon. Unlike a cockpit display, which is a fixed display that operates in Mimic mode, an instrument panel is an independent window (like an inset view) that you can place anywhere on the screen (Figure 490). It does not affect the observer mode. You can organize multiple instrument panels to create your own virtual cockpit.

Figure 490. Instrument panels



When you create an instrument panel, the title bar of the window lists the entity for which you created it.

- Instrument panels are GL Studio controls.
- Instrument panels are not supported on Linux.
- If the source of the instrument panel is removed from the exercise, for example when a Logger recording rewinds, the instrument panels are removed from the inset windows. To see them again you must open new instrument panels. Because the relationship between an instrument panel window and an entity is not persistent across multiple runs of an exercise, you cannot save an instrument panel layout for an entity. (You could save the layout of the windows to a display configuration, but they would not display instrument panels when you loaded the configuration.)
- If you want to put multiple instrument panels in one window, like the six-pack panel provided with VR-Forces, you must use GL Studio.

➤ **To open an instrument panel:**

1. Select the entity for which you want to display an instrument panel.
2. Choose **Object > Create Window > Create Instrument Panel >** *panel*, where *panel* is one of the instrument panels listed.

You can also right-click on the object and then choose **Create Window > Create Instrument Panel >** *panel*.

62.3.1 Configuring Entities to Use Instrument Panels

VR-Forces includes a set of instrument panels for fixed-wing aircraft, although many of them may be appropriate for rotary-wing aircraft, and even for ground vehicles. (For example, you might want to show a compass for a ground vehicle to display its heading, or an altimeter to display the height above sea level.) The availability of instrument panels to an entity is determined by the **instrumentPanelCategory** parameter in its model definition.

The instrument panels provided with VR-Forces have the **instrumentPanelCategory** set to FixedWing. All fixed-wing entities have this parameter set in their model definition. None of the other entities do. If you want other entities to display instrument panels provided with VR-Forces, you can add this parameter to their model definitions. (You could also copy an instrument panel model definition and change the category to some other value if you want to. Regardless, to use an instrument panel, an entity's model definition must include the **instrumentPanelCategory** parameter and have it set to a valid value.)

If you create new instrument panels, you must add instrument panel model definitions for them and specify a value for the **instrumentPanelCategory** parameter. Then you can add the parameter to the appropriate entity model definitions.

62.4. Displaying Sensor Views

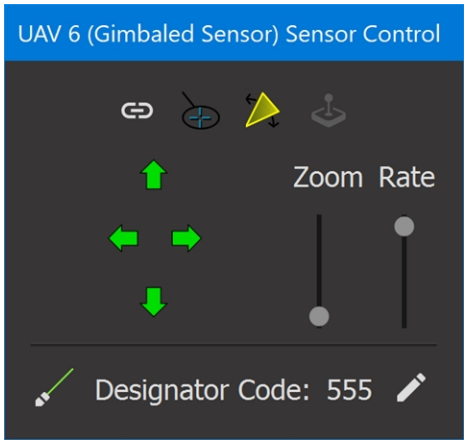
VR-Forces can display the view from gimbaled visual sensors, such as a camera on a UAV (3D only). The view displays in an inset window that has information about the observer mode and area being viewed (Figure 491). The window has its own observer and you can change the observer mode in the view.

Figure 491. Sensor View



Each sensor view has a control panel (Figure 492) that you can use to move the sensor, change its aim, and zoom in and out. You can also lase whatever the sensor is looking at.

Figure 492. Sensor Control Panel



The UAVSensorDemo example scenario demonstrates use of an entity with a gimbaled sensor. The scenario also has a Lua script that shows how to successively aim the sensor at many entities.

➤ **To display a sensor view:**

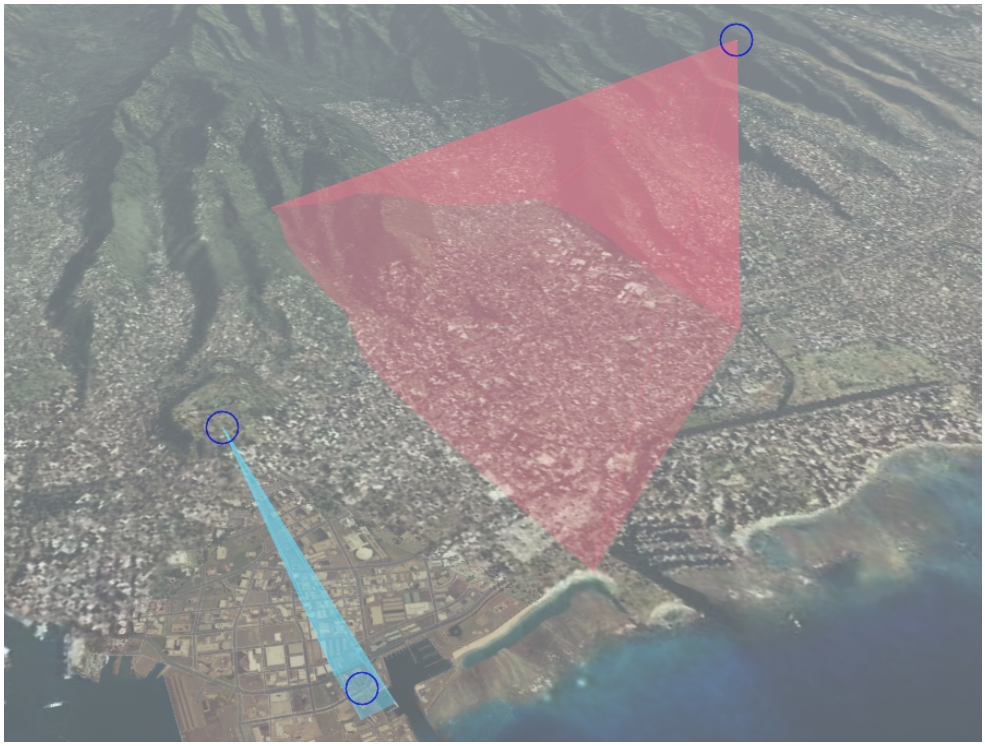
1. Select an entity that has a gimbaled sensor.
2. Choose **Object > Create Window > Create Sensor View >** *sensor*, where *sensor* is one of the sensors on the entity.

ⓘ You can also right-click on the object and then choose **Create Window > Create Sensor View >** *sensor*.

62.4.1 Displaying a Sensor Field of View Cone

You can display translucent cones that show the area on the terrain that a gimbaled sensor is observing. Each cone is a different color. Figure 493 shows two sensor field of view cones. One is focused on an entity and is zoomed in for a tight focus. The other is focused on a point on the terrain.

Figure 493. Sensor field of view cones



➤ **To display sensor field of view cones:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 448 on page 1165).
3. In the list of settings, select the Sensor Fields of View check box.

You can also enable and disable sensor field of view cones by choosing **Observer > Sensor Fields of View** or by clicking the Sensor Fields of View button ( ) on the Observer Settings Toolbar.

62.4.2 Controlling the Sensor View

When a sensor view opens, a control panel opens with it (Figure 492 on the previous page). This control panel works only with its associated sensor view. If you open multiple sensor views, there will be a control panel for each one. Use the Sensor Control Panel to:

- Aim the sensor at a simulation object or a point on the terrain. (You can also aim a sensor with the Sensor Aim set data request. For details, see "40.72. Sensor Aim" on page 899.)
- Set the sensor to scan mode. (You can also set a sensor to scan with the Sensor Aim set data request.)

- Take control of the sensor with a joystick. The joystick must be configured for use with the sensor view. For details about configuring joysticks, see "81.3. Configuring Joysticks and Keyboard Control" on page 1579.
- Zoom the sensor. (You can also zoom a sensor with the Sensor Zoom set data request. For details, see "40.74. Sensor Zoom" on page 899.)
- Move the sensor field of view. You can change the sensitivity of the movement buttons.
- Lase the center point of the sensor view.
- Set the laser's designator code.

If you zoom the sensor view to the maximum magnification and the terrain turns white, you can fix the problem by setting the clipping plane to Clamp Near Clip When Magnified. For details, see "83.4.1 Setting the Clipping Planes" on page 1605.

➤ **To aim the sensor at a simulation object:**

1. Click the Track an Entity button ( ). The Entity to Track dialog box opens.
2. Select the simulation object you want to aim at.
3. Click OK.

➤ **To aim the sensor view at a location:**

1. Click the Track Location button ( ). The Location to Track dialog box opens.
2. Click a location on the terrain or enter the coordinates in the dialog box.
3. Click OK.

➤ To set a sensor to scan mode, click the Scan button ( ).

➤ To control the sensor field of view with a joystick, click the Joystick button ( ).

➤ To zoom the sensor view, adjust the Zoom slider.

➤ To move the sensor view, click any of the arrow buttons.

➤ To change the rate at which the arrow buttons move the sensor view, adjust the Rate slider.

Controlling Lasing from the Sensor View

If the entity is equipped with laser guidance, you can enable lasing. If you enable lasing, the entity to which the sensor view is attached lases the center point of the sensor view (Figure 494).

Figure 494. Lasing a point in a sensor view



➤ To lase the center point of the sensor view, in the Sensor View control panel click the Lase Object button ( ) on the control panel.

VR-Forces calculates a unique laser code for each sensor view so that there are no duplicate codes. You can change the laser designator code.

➤ **To specify the laser designator code when lasing from a sensor view:**

1. In the Sensor View control panel, click the Laser Code button ( ). The Laser Code dialog box opens.
2. Type a 3- or 4-digit code between 111 and 1788, using only digits 1 through 8.
3. Click OK.

Configuring a Gimbaled Sensor's Azimuth

The characteristics of a gimbaled sensor, such as where it is located on an entity and the range through which the sensor can scan (the azimuth), are configured in the Gimbaled Visual Sensor system. You can edit many properties in the Simulation Object Editor. In particular, you may want to edit the azimuth. If you want the sensor to swivel 360 degrees, set the Minimum Azimuth Angle and Maximum Azimuth Angle to the same value.

➤ **To edit the properties of a gimbaled sensor:**

1. Open the Simulation Object Editor. (For details about using the Simulation Object Editor, see "Chapter 68. The Simulation Object Editor and SMSs" on page 1303.)
2. Select the entity that has a gimbaled sensor.
3. Select the Systems tab.
4. In the Sensors window, select the Gimbaled Visual Sensor.
5. Click the Edit Properties button ( ).
6. Edit properties as desired.

7. Click OK.
8. Save the SMS.

62.5. Using the Depth of Field Effect

Depth of field controls the area in the scene that is in focus. Depth of field is calculated based on a focal depth and a focal range. The focal depth is the distance from the observer that is the center (focal point) of the area in focus. The focal range is the distance from the focal point that is in focus.

In Figure 495, the focal depth is about 10 meters, fairly close to the observer, and the focal range is also small, so that the scene is out of focus just past the car (note that the trees are out of focus).

In Figure 496, the focal depth is the same, but the range is much greater, so the scene is in focus to the back of the parking lot but not the building.

In Figure 497, the focal depth is centered at the trees near the sidewalk. The focal range is, again, fairly narrow, so that the person and car in the foreground are out of focus.

VR-Forces lets you set the focal depth automatically or manually. In either case, you can adjust the focal range.

Figure 495. Depth of field (close)



Figure 496. Depth of field (far)



Figure 497. Depth of field, middle distance



➤ **To enable depth of field:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 373 on page 1052).
3. Select the observer mode for which you want to enable depth of field.
4. Select or clear the Depth of Field check box.

You can also enable or disable depth of field by choosing **Observer > Depth of Field**.

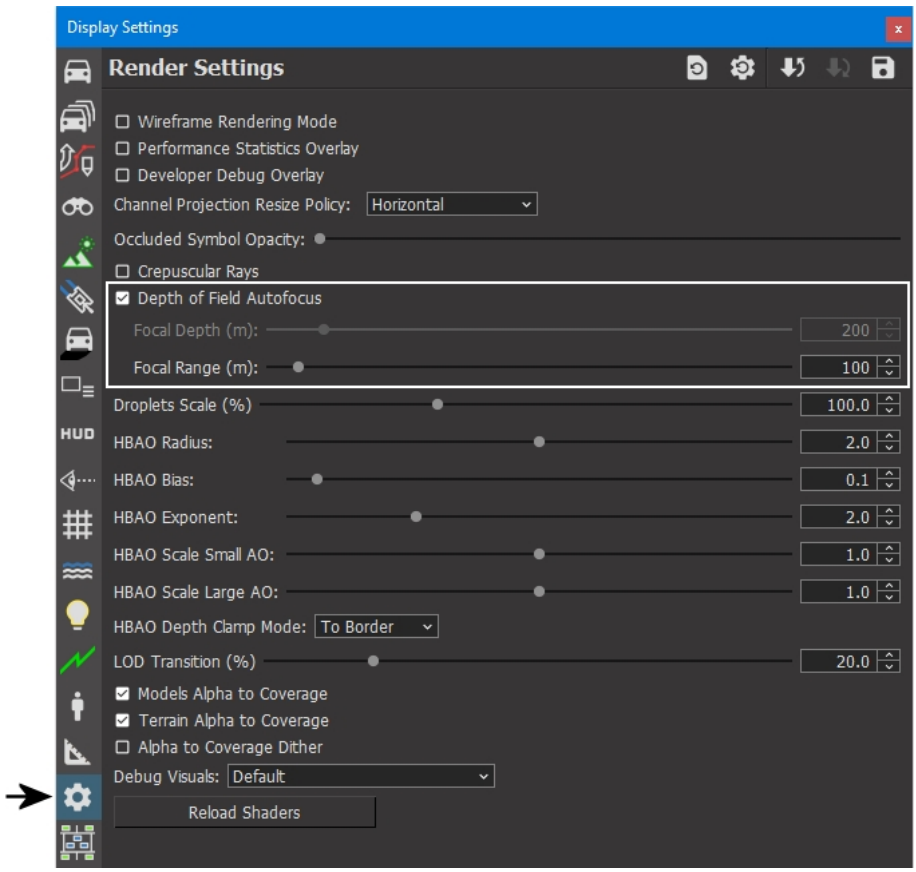
62.5.1 Configuring Depth of Field

You can enable and disable automatic depth of field and you can adjust the focal depth and focal range.

➤ **To configure depth of field:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 498).

Figure 498. Depth of Field settings



3. To use autofocus for focal depth, select the Depth of Field Autofocus check box.
4. To set the focal depth manually, clear the Depth of Field Autofocus check box and adjust the Focal Depth slider or specify a value in the box.
5. To adjust the focal range, adjust the Focal Range slider or specify a value in the box.



XII. Terrain

Each simulation takes place within the extents of a terrain database. A terrain database provides three-dimensional information about a geographic area and provides the environment within which simulation objects function. With the information in a terrain database, VR-Forces calculates height, intervisibility, and other three-dimensional information.

VR-Forces allows you to load terrain data in these basic ways:

- Compose a terrain from elevation, imagery, and feature data.
- Open an MTF file.
- Stream terrain from local files or a terrain server such as VR-TheWorld Online.
- Open a terrain database in a supported format, such as FLT, DTED, or GDB.

The recommended work flow is:

1. Create an earth file.
2. Connect to the earth file.
3. Save the terrain as an MTF file.
4. Create a scenario and choose the MTF file as the scenario database.

The decision making process for choosing a terrain when you create a scenario is described in "Chapter 7. Creating and Running Scenarios" on page 227.

Terrain Databases Provided with VR-Forces

Table 52 describes the terrain databases provided with VR-Forces. These terrains were created for you to use in your simulations. You may also use them to help create your own terrains. Some have variants that may be a different coordinate system, subset of the data, or other variation on the terrain. Terrains with "(online)" in the title connect to VR-TheWorld or another online server and therefore require an internet connection. Some terrains have variations that emphasize a particular set of features or other data. VR-Forces also has a supplemental data package that includes additional terrains not listed here. Please contact your MAK salesperson for download details.

Table 52. Terrain databases shipped with VR-Forces

Terrain Name	Description
Ala Moana.mtf	A high-resolution area of Honolulu, Hawaii, USA, set into a raster map of the world.
ArrivalsHall.mtf	An interior model of an international arrivals hall with passport control stations.
Bakersfield Topo Map (online).mtf	VR-TheWorld with topographical maps for the area around Bakersfield, California, USA. This terrain contains some feature data that simulation objects in aggregate-level scenarios can take advantage of.
Brooklyn.mtf Brooklyn (online).mtf Brooklyn with Vegetation.mtf	A high-resolution city center surrounded by an extruded cityscape. Located geographically in Brooklyn, NY, USA, but not an actual location in that city.
DestructibleTerrain.mtf	A small terrain for demonstrating destructible terrain based on switch nodes. The buildings also include switches for the doors and windows, allowing them to be opened or closed.
Example-ComposedTerrain.mtf	An example of a terrain created from pieces. Use this terrain and the associated earth file to help learn how to create your own.
Grid.mtf	A basic, flat terrain featuring only a simple grid design. This terrain is not suitable for VR-Forces scenarios.
Ground_DB II.mtf	FBX. A simple, generic ground terrain, including a road, a runway, and some buildings, surrounded by water.
MAK Earth (online).mtf MAK Earth (online) - Simple For High Fliers.mtf	VR-TheWorld terrain with high resolution insets for California, Honolulu, Hawaii, and some other cities. Includes streaming feature data. The simple version is intended for use when the observer will be high in the air. It is a more basic version of the MAK Earth (online) terrain and has improved performance for simulations where high-resolution ground details are not needed.

Table 52. Terrain databases shipped with VR-Forces (continued)

Terrain Name	Description
MAK Earth Aggregate (online).mtf	VR-TheWorld terrain designed for use in aggregate-level scenarios. When in 2D projection (plan view), you can choose the appearance from the 2D View Layer list. The Topo View layer looks like a paper map, whereas the default view resembles satellite imagery.
MAK Earth Air and Space (online).mtf	US Sectional Air Charts for IFR (Instrument Flight Rules) with base imagery. This terrain is designed for airspace simulation using the 2D projection (plan view), with simple lines and labels indicating countries. The VFR and and LL IFR 2D view layers show different information for the U.S.A. and its territories.
MAK Earth Base (online).mtf	VR-TheWorld MAK Earth terrain with high resolution insets for Honolulu, Hawaii, and Boston, Massachusetts. Includes streaming feature data.
MAK Proving Ground.mtf	A non-realistic terrain with buildings, trees, roads, a runway, different types of soil, slopes, and other environments. It is designed specifically for testing entity models.
NTC (online).mtf	A high-resolution area of the National Training Center at Fort Irwin in California, U.S.A.
PacificGroveTrail.mtf	A procedurally generated earth merged with high-resolution imagery and elevation. It includes dense vegetation to simulate a fictional forested area in San Diego county.
Sadr City.mtf	An urban geotypical Middle Eastern terrain in Sadr City, Iraq, a suburb of Baghdad. Ideal for urban scenarios.
Test terrains	Multiple terrains used for testing models and different terrain database capabilities. They include a terrain with S-57 chart data, procedural terrains, as well as flat earth and geocentric terrains. Not all test terrains are suitable for VR-Forces scenarios, for example, Test-WorldFlatEarthNoElevation.
Truman Carrier.mtf	A model in VR-Forces of an aircraft carrier as a terrain object rather than a simulation object.
VR-Village.mtf	GDB. Feature data. OpenFlight terrain. VR-Village is a fictional database created by MAK. It has a highly detailed small town in a larger, sparsely detailed desert environment. The town includes multilevel buildings, underground tunnels, and a road that tunnels through a mountain. The coordinates place it off the west coast of Africa.



63. Introduction to Terrain Usage in VR-Forces

This chapter provides introductory information about terrain usage, including supported file formats, coordinate systems supported, and use of large terrain databases.

63.1. Composing Terrains (and Creating MTF Files) 1242
63.2. Supported File Formats 1242
Terrain Data 1243
Elevation Data 1243
Supported Imagery Formats 1243
Supported Feature Data Formats in the GUI 1244
63.3. Feature Layers and Props 1244
63.3.1 Geometric Representation of Point Feature Data 1244
63.3.2 Feature Data Types Supported by the Simulation Engine 1245
63.4. Coordinate Systems 1245
63.4.1 The UTM Coordinate System 1245
63.4.2 Geocentric Terrain Databases 1246
63.5. Loading Supported Databases 1246
63.6. Using Large Terrain Databases 1246
63.6.1 Terrain Paging 1247
63.6.2 Terrain Streaming 1248
63.7. Using Ground Clamping 1248
63.7.1 Configuring Ground Clamping 1249

63.1. Composing Terrains (and Creating MTF Files)

VR-Forces lets you compose terrains from a variety of supported data formats. You can combine elevation data (terrain patches), imagery, and feature data to create terrains.

VR-Forces offers some basic methods for creating terrains:

- Compose a terrain by loading individual elevation, image, and feature data files. This method is best for relatively small terrains that do not require paging. It supports all of the coordinate systems that VR-Forces supports. For more information, see "Creating a Composed Terrain" in the *Adding Content to MAK ONE Applications Reference Manual*.
- Specify a set of files to load using an osgEarth format earth file. Use this method to stream terrain data from local or remote sources. This method is best for terrains with large amounts of elevation or image data, which would benefit from paging. For more information, see the topic "Connecting to Terrain Servers" and also the chapter *Loading Data Using Earth Files*, both in the *Adding Content to MAK ONE Applications Reference Manual*.

When you compose a terrain, you can save it as a MAK terrain file (MTF), which is essentially a description of all of the data used to visualize the simulated 2D and 3D environment. Saving a terrain lets you automatically load all of the data that you loaded manually when you first created it.

You can save terrains in a cached format that loads terrain data more quickly than if the terrain is created from the source data. For details, see "Configuring File Caching" in the *Adding Content to MAK ONE Applications Reference Manual*.

63.2. Supported File Formats

VR-Forces can import elevation, terrain, images, models, and feature data. Some types of files can contain more than one type of information. For example, GeoTIFF files can contain elevation data, imagery, or land cover data. The following sections list the supported formats.

> In theory, VR-Forces can load any format supported by osgEarth if you use the earth file format to load them. However, VR-Forces does not include drivers for all of the formats supported by osgEarth. Furthermore, when using osgEarth, VR-Forces supports only geocentric terrains. If you are uncertain about support for a particular format, please contact your MAK distributor or use the Support Portal at https://www.mak.com/mak-one/support/portal.

Terrain Data

VR-Forces imports terrain data directly as terrain patches. Supported formats include:

- MAK encrypted data format (*.medf*)
- OpenFlight (*.flt*)
- MetaFlight (*.mft*)
- TerraPage (*.txp*)
- CDB (using an earth file)
- GIS source data formats (using an earth file) including imagery, elevation, and features

VR-Forces provides limited support for data in these formats. If you have problems loading data in these formats, MAK will try to assist you but cannot guarantee that it will be possible to use your data.

- 3DS
- OSG
- FBX

Elevation Data

VR-Forces supports these types of elevation data:

- DTED (*.dt0*, *.dt1*, *.dt2*, *.dted*)
- GeoTIFF
- DEM
- Arc/Info Binary Data (*.adf*)
- Other formats supported by osgEarth

Supported Imagery Formats

You can load imagery in these formats:

- GeoTIFF (*.tif*, *.tiff*)
- JPEG2000 (*.jpg2*)
- MBTiles (Raster)
- ECW (Windows only)
- Other formats supported by osgEarth using the GDAL driver. For a list of formats supported by GDAL, go to https://gdal.org. If you have problems loading a format that is not in this list, please ask on the Support Portal at https://www.mak.com/mak-one/support/portal or contact your distributor.

Although many formats and properties are supported, for optimal performance, imagery or elevation files should:

- Use the WGS-84 (EPSG:4326) Coordinate Reference System (CRS).
- Be tiled.
- Have overviews.

An ideal format is Cloud Optimized GeoTIFF (COG).

We cannot guarantee that unlisted formats are fully supported. For more information about earth files, see "Connecting to Terrain Servers" and the chapter on *Composing a Terrain from Data* in the *Adding Content to MAK ONE Applications Reference Manual*.

Supported Feature Data Formats in the GUI

VR-Forces supports these feature data formats in the GUI:

- Shape files
- S-57
- FlatGeobuf
- OGR vector formats referenced in earth files

63.3. Feature Layers and Props

A terrain can have objects such as buildings and trees embedded in it. These objects cannot be manipulated in this format. However, VR-Forces lets you extract these types of objects from the terrain and render them independently as props. Props are terrain elements that you can manipulate as separate objects.

You can add props to a terrain by extracting them from an external reference in a terrain patch or from a feature layer. You can also add props directly from the Props Palette.

A feature layer is a file that contains feature data such as point features, linear features, and areal features. You can extract point features as props. You can extract props from MAK Terrain Format (GDB) files, shapefiles (SHP), or DFAD data (DFD).

For more information, see "Adding Props to a Terrain" in the *Adding Content to MAK ONE Applications Reference Manual*.

63.3.1 Geometric Representation of Point Feature Data

There are differences between how the GUI and sim engine process feature data. This is because the information needed to simulate simulation object interaction with features is different from the information needed to render the scene.

The GUI renders point features such as buildings or trees based on the data provided in the feature layer. These objects could have been extracted as props, in which case they are represented as geometric objects in the terrain.

If you are streaming features using an earth file, you can configure the earth file to extrude buildings into 3D geometry. This capability is demonstrated in some of the insets in *MAK Earth (online).mtf*, for example the Boston and Honolulu insets.

The simulation engine's representation of point features depends on how they are used. Collision avoidance treats them as bounding volumes with infinite height. Line-of-sight calculations treat them as bounding volumes with a defined height. If you extract features as props, the sim engine treats them as geometry where that is appropriate and as features otherwise.

The sim engine does not recognize extruded geometry.

63.3.2 Feature Data Types Supported by the Simulation Engine

The sim engine supports the following types of feature data files:

- MAK terrain database (*.gdb*)
- Shape terrain data (*.shp*)
- S-57
- WFS
- OGR vector formats referenced in earth files
- VMAP

The preferred format for adding features is shapefiles. You can extract feature data from GDB, DFD, and VMAP into shapefiles with the vectorNetworkToShapeFiles utility. For details, see "Extracting Feature Data from GDB to Shapefiles" in the *Adding Content to MAK ONE Applications Reference Manual*. You can extract feature data from S-57 files into shapefiles with the ogr2ogr tool. For details, see "Converting S-57 Feature Data to Shapefiles" in the *Adding Content to MAK ONE Applications Reference Manual*.

63.4. Coordinate Systems

VR-Forces supports UTM, geocentric, flat earth, and Lambert Conformal Conic coordinate systems. There is no presumption that a particular database format, such as OpenFlight or DTED, will use a particular coordinate system.

63.4.1 The UTM Coordinate System

VR-Forces includes several databases that use Universal Transverse Mercator (UTM) coordinates. UTM databases are well-suited to two-dimensional views and simulations on a relatively small geographic area. However, their accuracy decreases as you move away from the origin, so they are not well-suited for exercises that need to simulate over areas greater than one UTM zone.

There are several types of UTM systems. In general, the earth is divided into 60 zones generally 6 degrees wide in longitude. These zones may be further subdivided into quadrangles by using letter designations from south to north. This complicates references to UTM. For example, standard offset could be used with or without the quadrangles (which affects the northing portion of the coordinate), non-offset does not use the quadrangles, and Milgrid requires the quadrangles.

More specifically, UTM coordinates define two dimensional, horizontal, positions. UTM zone numbers designate 6 degree longitudinal strips extending from 80 degrees South latitude to 84 degrees North latitude. UTM zone characters designate eight degree zones

extending north and south from the equator. There are special UTM zones between 0 degrees and 36 degrees longitude above 72 degrees latitude and a special zone 32 between 56 degrees and 64 degrees north latitude.

VR-Forces uses this definition:

A non-Cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth. When in UTM mode (the default), VR-Forces uses an offset UTM coordinate system, one in which coordinates are expressed with respect to an origin located at the latitude and longitude specified in a configuration file. Typically, this will be the location of the origin of your terrain database.

63.4.2 Geocentric Terrain Databases

To support large play areas, simulations can use geocentric coordinates. In the geocentric coordinate system, the origin is at the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the North Pole.

63.5. Loading Supported Databases

MTF files reference elevation, image, and feature files that VR-Forces processes and loads when you load a scenario. However, you can also load terrain databases directly into VR-Forces. They will be converted to the VR-Forces internal format using default values. This process is less efficient than loading MTF terrains, may take more time, and must be repeated each time you load the terrain as part of a scenario or in the GUI.

When you create a scenario, you can choose a terrain in any supported format. If you want to open a terrain using the **File > Open Terrain** menu option, you can only choose MTF files. To load a non-MTF terrain out of the context of creating a scenario, you must load it as a terrain patch.

If you load a terrain directly, you can still save it to MTF.

63.6. Using Large Terrain Databases

VR-Forces can support large databases, where large might mean a high polygon count, a large geographic area, or both. VR-Forces supports two methods for dealing with large databases – terrain paging and streaming terrain. Both approaches have a common goal – providing access to large amounts of data without loading all of that data into memory at any one time.

VR-Forces cannot arbitrarily page or stream terrains. It can page terrains that have been designed for paging and can stream terrains that are accessed through a terrain server, such as VR-TheWorld. For details about support for paged terrains, see "Loading MetaFlight Terrains" in the *Adding Content to MAK ONE Applications Reference Manual*. For details about connecting to terrain servers, see "Connecting to Terrain Servers" and "Creating an Earth File to Load Terrain Data", also in the *Adding Content to MAK ONE Applications Reference Manual*.

> Even if you use terrain paging or streaming terrain, it is still possible that a large terrain could exceed the memory limitations of your operating system.

63.6.1 Terrain Paging

Terrain paging breaks up a database into multiple cells that can be loaded independently. By comparison, a non-pageable terrain must be loaded in its entirety. When VR-Forces loads a paged terrain, the GUI loads the terrain at whatever level of detail is required by the zoom level. The simulation engine always loads terrain at the highest level of detail. Therefore, if you rapidly zoom into a section of the terrain in the GUI, there may be a delay in correlation between simulation objects and the terrain until the GUI catches up to the proper level of detail.

In general, the simulation engine does not load terrain until it needs the terrain to support the objects in the scenario. (Some terrain data is always loaded immediately, such as terrain data in OpenFlight format.) For example, if you create a new scenario and specify a paged terrain, the simulation engine does not load terrain until you create a simulation object. Then it only loads the terrain page required for the object.

When you run a scenario, as simulation objects move, the simulation engine tries to predict which terrain pages it will need to support the simulation objects and tries to page them in before the simulation objects need them. If the total polygon count required to page in required terrain exceeds the maximum polygon count configured, VR-Forces pages out (removes) unused pages. The simulation engine pages out the least recently used data.

> Predictive terrain paging is done in real time, not just when a scenario is running. For example, when you load a scenario, all required terrain is paged in and when you add objects during scenario creation, terrain is paged in.

You can also manually page in terrain using terrain page-in areas. For details, see "Manually Paging-In Terrain in VR-Forces" in the *Adding Content to MAK ONE Applications Reference Manual*.

Paging is done based on these priorities:

- If a simulation object is moving, paging in terrain that it needs is a higher priority than predictive paging.
- Blocking terrain calls have a higher priority than non-blocking calls. (A blocking call is one that prevents the simulation from progressing until the call is completed. For more information, see the **assertOnBlockingTerrainCalls** parameter in Table 76, in "C.1. The vrfSim.mtl Configuration File Parameters" on page 1668.)
- Paging out terrain to recover memory is a higher priority than paging in terrain. The need to page out is calculated based on the polygon count of the terrain, not on its memory usage.

63.6.2 Terrain Streaming

The term terrain paging is used in reference to static, local terrains. Another way to load terrain data for a large area is to stream it from a terrain server. The idea is similar to terrain paging. Only the data you need at a particular location is loaded. The difference is that a terrain server can serve many computers at a time and normally holds much more data than a static local terrain. It can stream elevation data, imagery, map data, and feature data. It may have data at multiple levels of resolution for a particular portion of the terrain. VR-Forces supports connection configurations to terrain servers, including MAK's VR-TheWorld Server.

63.7. Using Ground Clamping

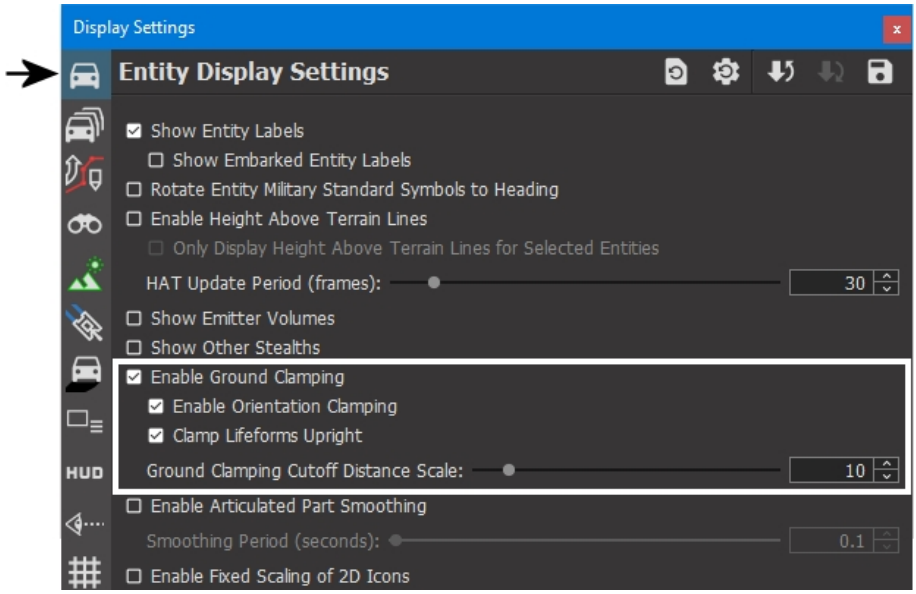
Because of differences in terrain databases among exercise participants, entities can sometimes appear to be underground or hovering above the terrain surface. To compensate, VR-Forces can clamp land and sea entities to the terrain surface. When ground clamping is enabled, VR-Forces keeps an entity anchored to the terrain surface regardless of the altitude data contained in its state update.

- Use of ground clamping can reduce your frame rate. If you correlate the simulation and visualization terrain databases, you can eliminate the need to use ground clamping.
- Although ground clamping controls are available in aggregate-level scenarios, since aggregate-level scenarios do not use 3D models, they are not useful.

➤ **To enable or disable ground clamping:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 499).

Figure 499. Ground clamping settings



3. Select or clear the Enable Ground Clamping check box.

> You can also enable and disable ground clamping by choosing **Settings > Ground Clamping**.

63.7.1 Configuring Ground Clamping

You can configure aspects of ground clamping:

- Enable or disable ground clamping.
- Enable or disable orientation clamping.
- Force lifeforms to remain upright relative to the plane of their location when moving up slopes or steps, rather than being perpendicular to the terrain.
- The distance from the terrain within which entities are clamped.

➤ **To configure ground clamping:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Entity Display Settings page (Figure 499 above).
3. To enable or disable a ground clamping option, select or clear its check box.
4. To specify the distance from the terrain in which entities are ground clamped, adjust the Ground Clamping Cutoff Distance Scale slider or type a value, in meters, in the box.

> The setting for the Ground Clamping Cutoff Distance Scale maximizes at 100 meters. If you set it to the maximum, the distance is infinite and all entities are ground-clamped.



64. Displaying Terrain Information

This chapter describes how to change the display of terrain to provide different types of useful terrain information.

64.1. Introduction 1252
64.2. Displaying Grid Lines 1252
64.2.1 Configuring Grid Lines 1253
64.3. Displaying Models and Terrain in Wireframe Mode 1254
64.4. Filtering Earth File Layers 1255
64.5. Displaying Feature Data (2D) 1256
64.5.1 Displaying a List of Vector Features 1258
64.5.2 Changing the Colors Used for Feature Data 1259
64.6. Adjusting Map Colors 1260
64.6.1 Adjusting Map Colors on the Display Settings Toolbar 1262
64.7. Selecting a 2D View Layer 1262
64.8. Displaying Coordinates and Soil Type for a Location 1262
64.9. The Map Scale Overlay 1263
64.10. Configuring Visible Surfaces (GDB Soil Types) 1263

64.1. Introduction

As you create and view simulations, you may need information about the terrain. By changing how terrain is displayed, you can get some of this information. If your terrain does not have texture data or an image overlay, features such as elevation coloring can provide a more realistic look to the terrain.

For information about how to compose terrains, see the chapter *Composing Terrains* in the *Adding Content to MAK ONE Applications Reference Manual*.

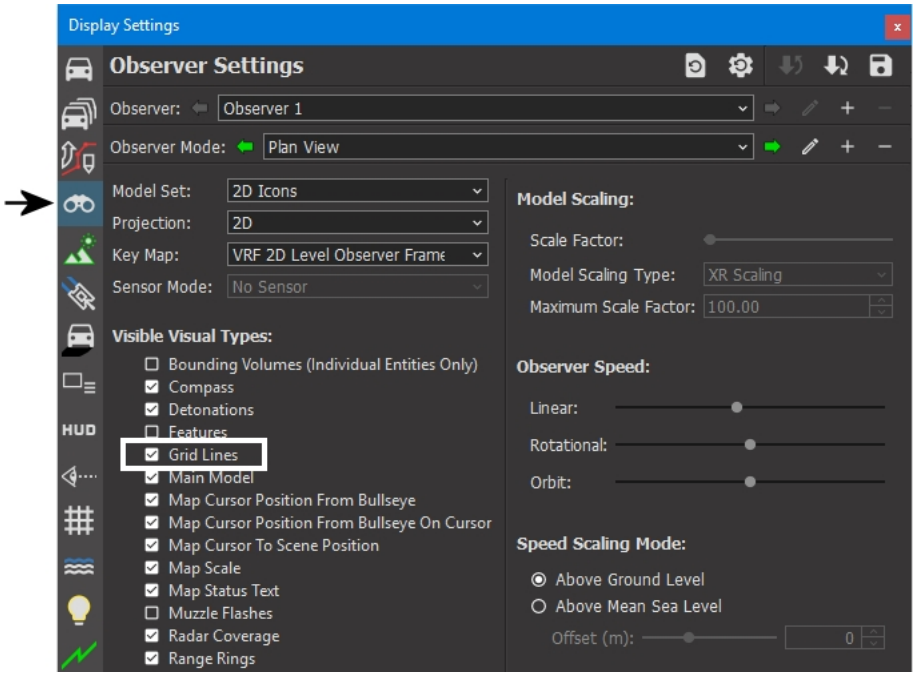
64.2. Displaying Grid Lines

VR-Forces can display grid lines in the 2D view. Grid lines are an observer-specific setting. They are configurable, as described in "64.2.1 Configuring Grid Lines" on the facing page.

To enable or disable grid lines:

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 500).

Figure 500. Observer Settings page



3. Select Plan View mode (or a locally created 2D observer mode).
4. Select or clear the Grid Lines check box.

> You can also enable or disable grid lines by choosing **Observer > Grid Lines**.

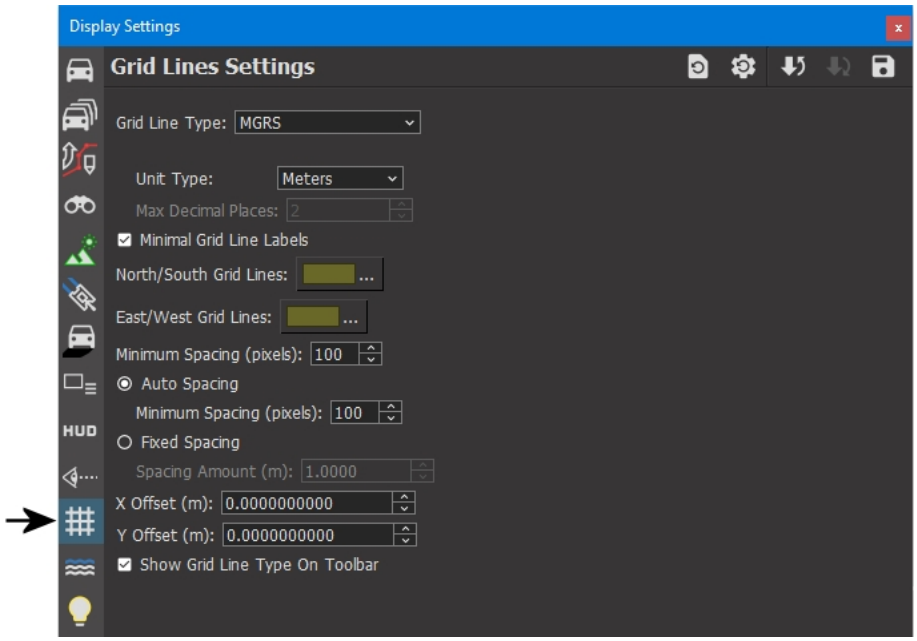
64.2.1 Configuring Grid Lines

You can configure the coordinate system used for grid lines, the precision, and the spacing. Spacing can be fixed, in meters, or determined by VR-Forces using a specified number of screen pixels. Fixed spacing can result in a very tight grid and unreadable labels as you zoom out. Auto spacing does not have this problem. However, the grid lines constantly recalculate as you zoom in and out.

➤ **To configure grid lines:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Grid Lines Settings page (Figure 501).

Figure 501. Grid Lines Settings page



3. To change the coordinate system used for grid lines, select an option in the Grid Line Type list. The list displays grid line types that are appropriate for the type of database loaded. (Until a database is loaded, the settings are blank.)

> You can also select the grid line coordinate system using the Grid Line Selector Toolbar.

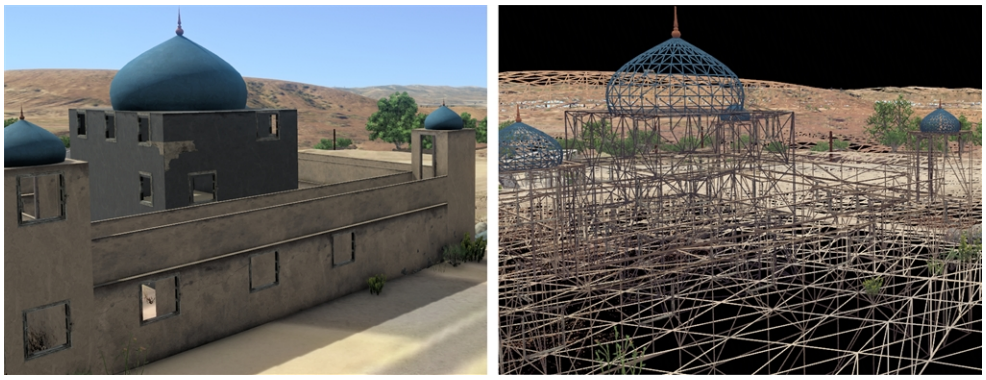
4. To specify the unit type for the grid line labels, select an option on the Unit Type list.
5. To specify the precision of the grid line labels, enter a value in the Max Decimal Places box.
6. To show more or less information at each grid line, clear or select the Minimal Grid Line Labels check box, respectively.

7. To use different colors for the north/south or east/west grid lines, choose a color for the lines.
8. To change the spacing type, select AutoSpacing or Fixed Spacing and specify a value for the chosen spacing type.
9. To specify an offset from the origin, change the values for X Offset and Y Offset.
10. Specify the altitude at which VR-Forces should stop putting latitude and longitude coordinate labels on the sides of the window and instead place them in the center of the world. The default is when the observer viewpoint moves into low earth orbit.
11. To include the coordinate system on the Grid Line Selector toolbar, select the Show Grid Line Type On Toolbar check box.

64.3. Displaying Models and Terrain in Wireframe Mode

In wireframe mode, VR-Forces displays the polygons that make up the terrain and objects (Figure 502).

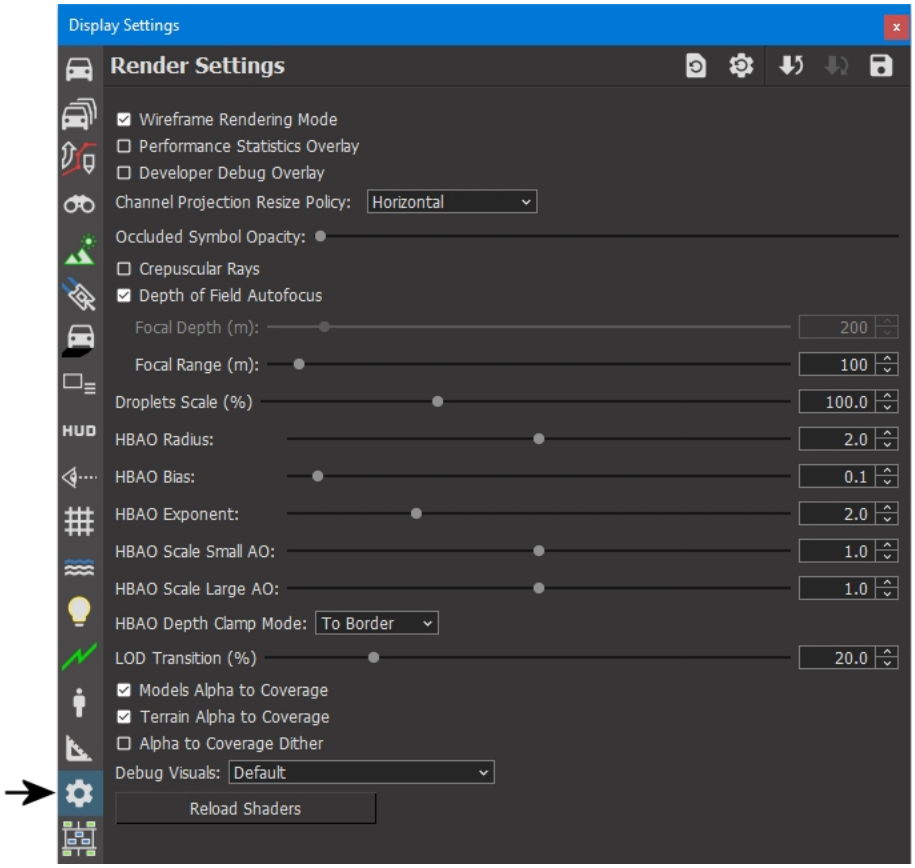
Figure 502. Wireframe off and on



➤ **To enable or disable wireframe mode:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 503).

Figure 503. Render Settings page



3. Select or clear the Wireframe Rendering Mode check box.

> You can also enable and disable wireframe mode by choosing **Settings > Wireframe Rendering**.

64.4. Filtering Earth File Layers

When you load a terrain using an earth file, imagery and feature data is specified in image elements and model elements. You can load multiple instances of these types of data, resulting in a rich, complex terrain. However, if you are only interested in certain types of terrain features, such as roads or rivers, displaying all of the terrain data can obscure the data that you are most interested in. To help declutter the map, you can filter out image and model layers globally or by observer mode.

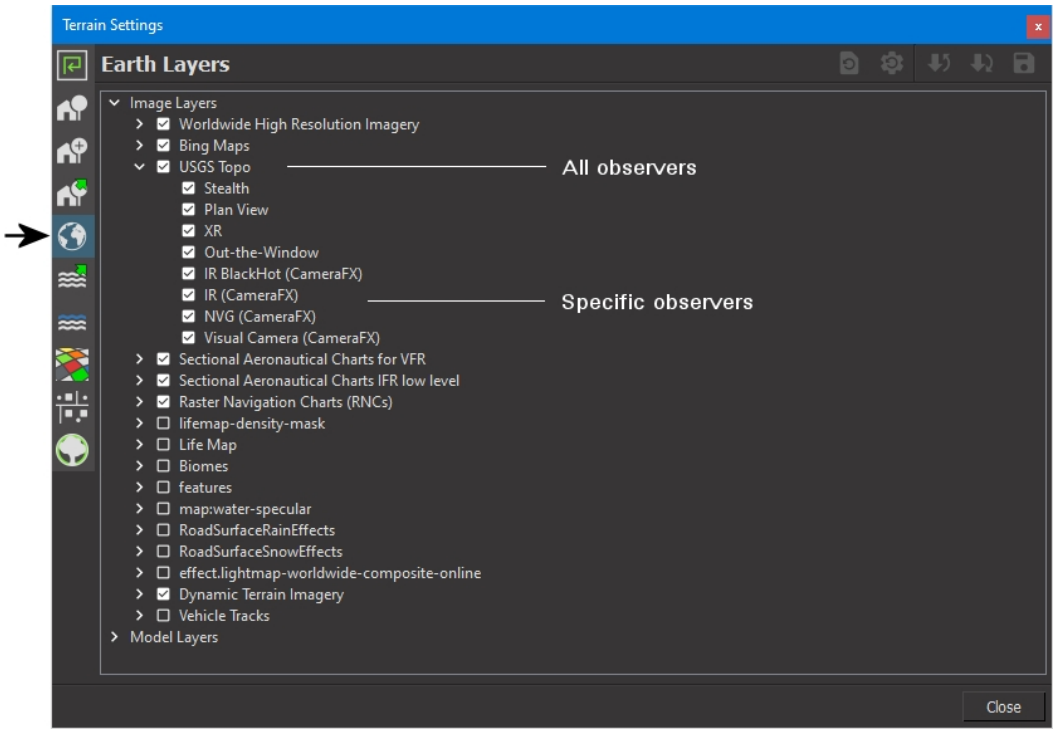
This feature is particularly useful if you want to show some layers in the PVD observer mode and others in one of the 3D observer modes. Settings are saved to the MTF file so that they are in effect in all scenarios that use the terrain.

- Filtering an earth file layer affects all open windows. You cannot hide a layer in one window and leave it open in a different one.
- Filtering only affects the GUI. All layers are loaded in the sim engine.

➤ **To filter out earth file layers:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Earth Layers page (Figure 504).

Figure 504. Earth Layers page



3. To display or hide an image or model layer for all observer modes, select or clear its check box.
4. To display or hide an image or model layer for a particular observer mode, expand the entry for the layer you want to change and select or clear the check box for the observer mode you want to change.

64.5. Displaying Feature Data (2D)

If a terrain contains feature data imported from GDB files or shapefiles, you can display it in the 2D view observer mode. Figure 505 shows a section of the Little Pond terrain with features disabled and enabled.

Figure 505. Feature data



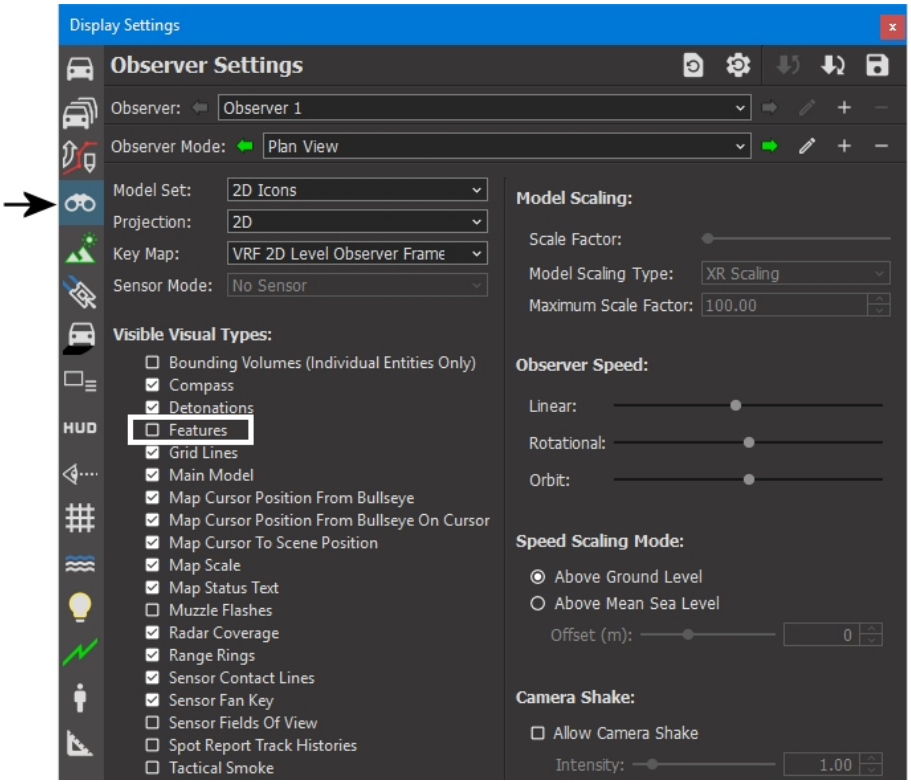
This setting only displays features imported from GDB files or shapefiles. It does not display features streamed from a terrain server. If you load features using an earth file, they are drawn (or not drawn) based on the configuration information in the file.

If your terrain has many features, displaying them can take a long time.

➤ **To show or hide feature data:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page.
3. In the Observer Mode list, select Plan View (Figure 506).

Figure 506. Observer Settings page, Plan View observer mode



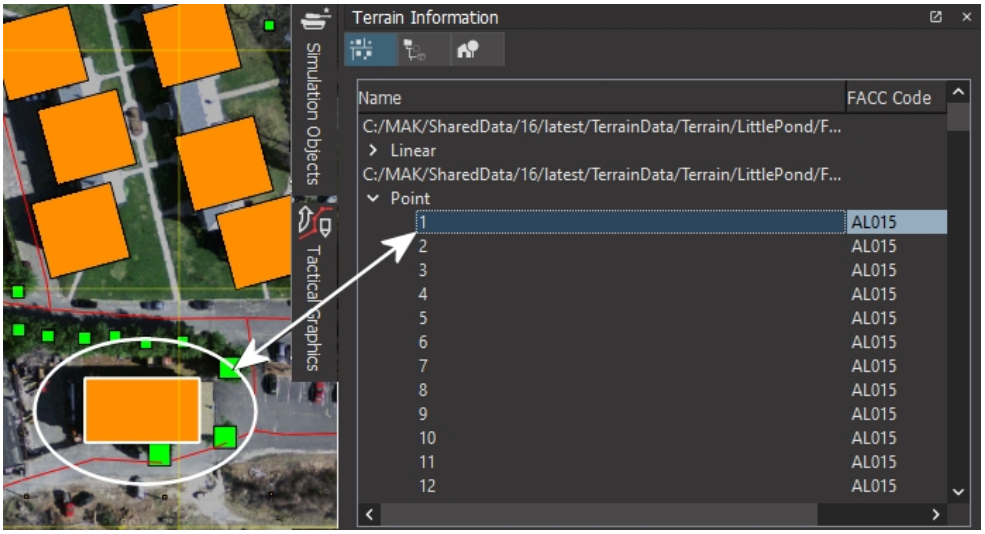
4. Select or clear the Features check box.

You can also enable or disable display of feature data by choosing **Observer > Features**.

64.5.1 Displaying a List of Vector Features

If a terrain has vector features, they are listed on the Vector Network tab of the Terrain Information Panel. If you select a feature in the list, it is highlighted in the display, and the other way around (Figure 507).

Figure 507. Vector feature selected



➤ **To display the feature list:**

1. Choose **View > Terrain Information Panel**. The Terrain Information Panel opens.
2. Select the Vector Network tab (Figure 507).
3. Expand the top-level categories to see the lists of individual features.

64.5.2 Changing the Colors Used for Feature Data

If a terrain database contains feature data, you can change the colors used to display it. In Figure 508, the Woodland feature in the Little Pond terrain was changed to pink. Compare it to the default color (Figure 505 on page 1257).

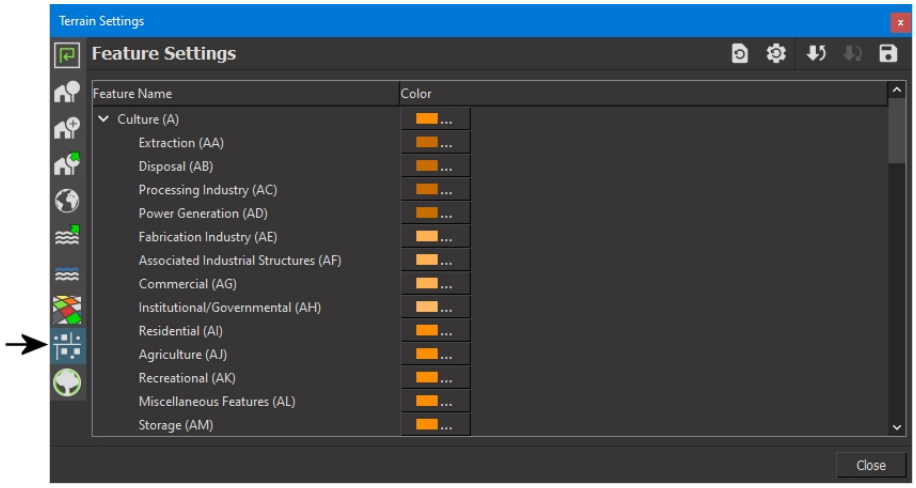
Figure 508. Feature data with color change



➤ **To change the color of feature data:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Feature Settings page (Figure 509).

Figure 509. Feature Settings page

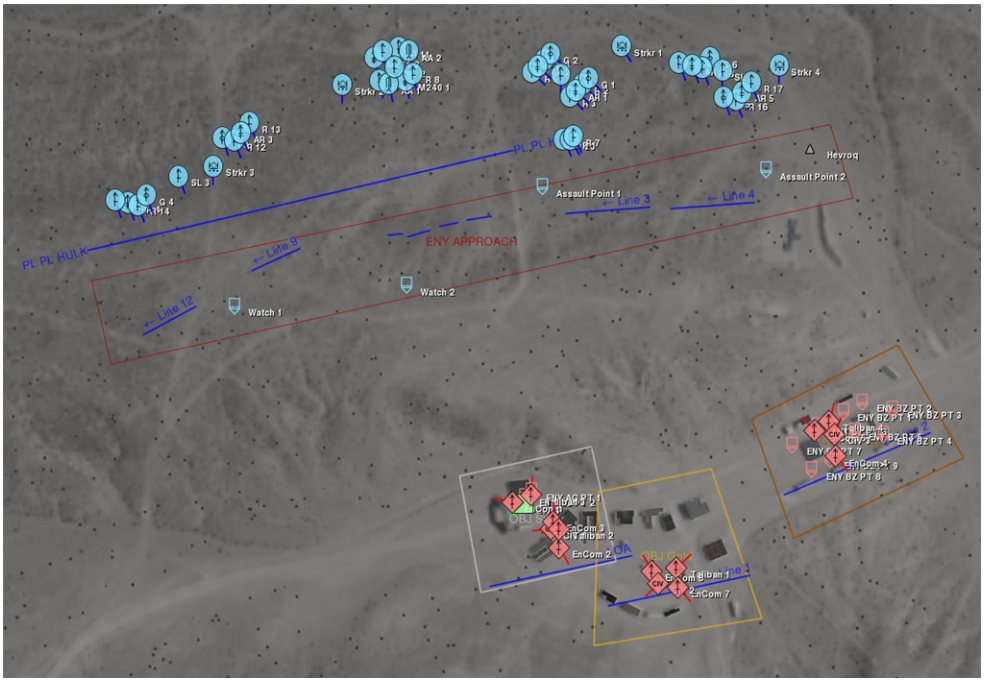


3. Click the color swatch for the feature that you want to change. A color picker dialog box opens.
4. Select the color that you want to use.
5. Click OK.

64.6. Adjusting Map Colors

In the 2D view, you can change how the colors are displayed. You can adjust the contrast, brightness, and color mix. You can also invert the grayscale. This allows you to view the scenario without the colors of the terrain distracting from what you want to see.

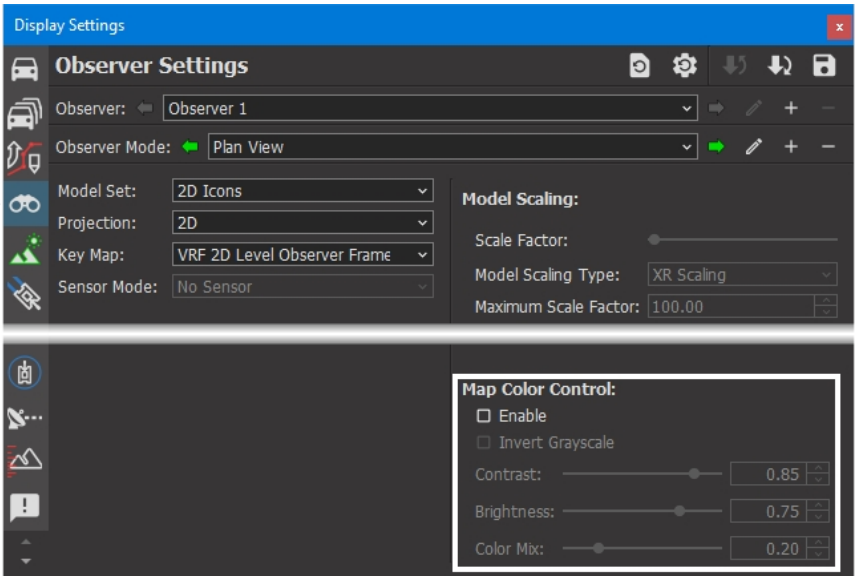
Figure 510. Map color set to grayscale



➤ **To adjust the terrain colors:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Observer Settings page (Figure 511).

Figure 511. Map Color Control Settings



3. Under Map Color Control, select the Enable check box. The terrain colors are changed to reflect the settings.

4. Optionally, select or clear the Invert Grayscale check box.
5. Adjust the Contrast, Brightness, or Color Mix values as desired.

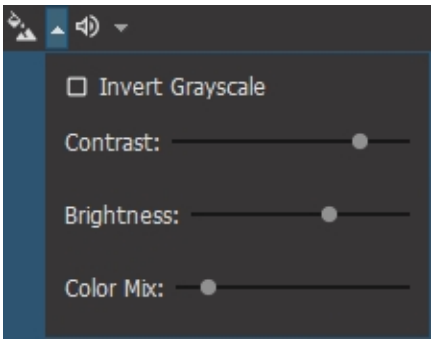
64.6.1 Adjusting Map Colors on the Display Settings Toolbar

You can also manage the Map Color Control using the button on the Display Settings toolbar.

➤ **To enable or disable map color control on the Display Settings toolbar:**

1. Click the Map Color Control button () to enable or disable the settings.
2. Click the arrow to display a small panel with the available settings (Figure 512). It is a persistent window.

Figure 512. Enter Caption Here



3. Set the values as desired.
4. To hide the panel, click the arrow on the Map Color Control button.

64.7. Selecting a 2D View Layer

osgEarth terrains can have different visible layers to display in the 2D view. These can be layers defined specifically to display contour lines, annotations, and so on.

➤ To select a 2D View Layer, choose the layer from the 2D View Layer toolbar.

64.8. Displaying Coordinates and Soil Type for a Location

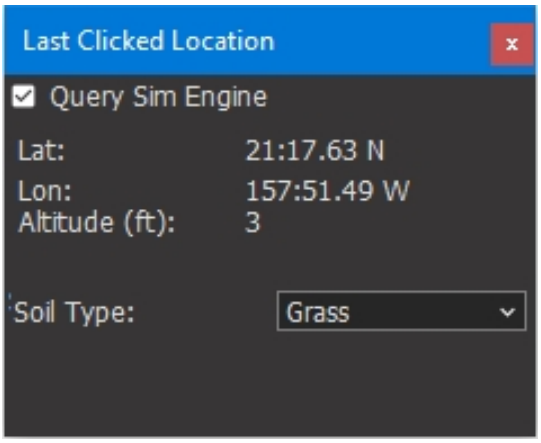
You can display the coordinates and soil type (if defined) of features and any point in the terrain. Soil type must be provided by the simulation engine. If you do not choose to query the simulation engine, the Last Clicked Location Panel just displays coordinates using data from the GUI.

The Last Clicked Location Panel becomes unavailable if you query the simulation engine and it takes more than 1/2 second for a response.

➤ **To check the coordinates and soil type of a feature or location:**

1. If the Last Clicked Location Panel is not open, choose **View > Last Clicked Location**.

Figure 513. Last Clicked Location Panel



2. If you want soil type information, select Query Sim Engine.
3. Click a point on the terrain or click a feature. Information appears in the Last Clicked Location Panel.

64.9. The Map Scale Overlay

In Plan View Observer mode, you can display a map scale overlay on the terrain.

Figure 514. Map scale overlay

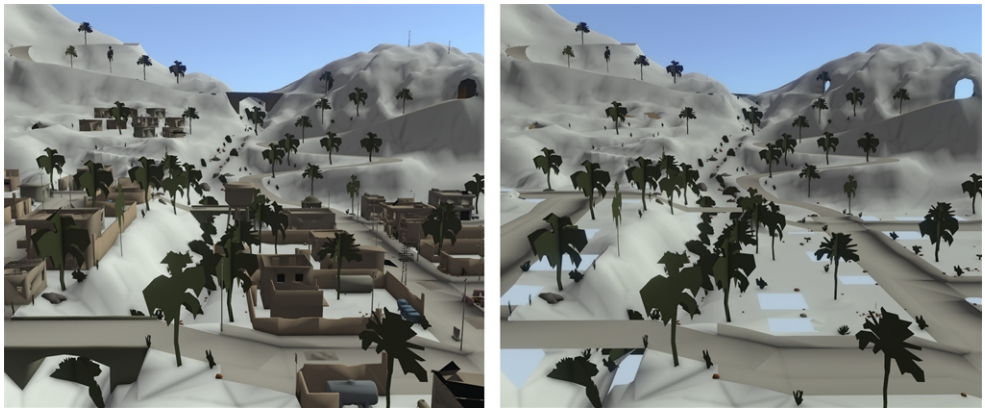


➤ To view or hide the map scale overlay, choose **Observer > Map Scale**.

64.10. Configuring Visible Surfaces (GDB Soil Types)

If a terrain patch was loaded from a GDB file, you can display or hide GDB soil types. Figure 515 shows the *VR-Village.gdb* terrain file. The view on the left shows all soil types. The view on the right has Building, Grass, and Rock surfaces hidden.

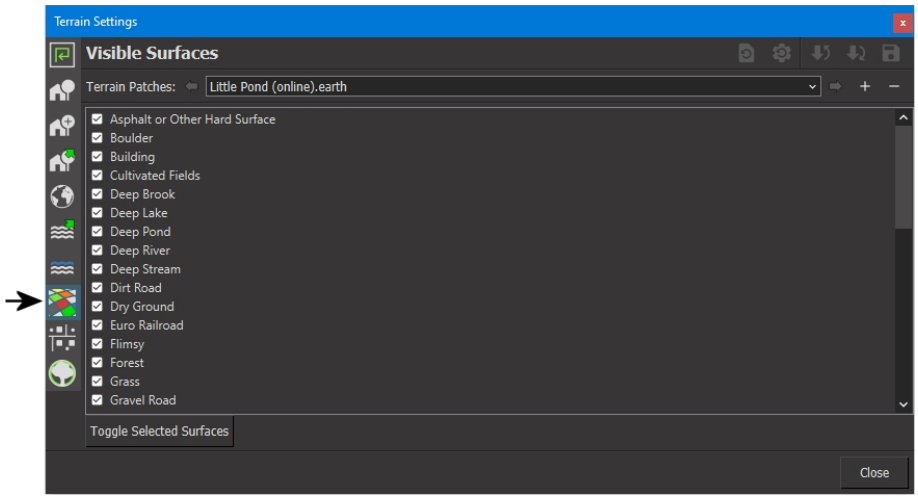
Figure 515. Visible surfaces toggled off



➤ **To display or hide GDB soil types:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Visible Surfaces page (Figure 516).

Figure 516. Visible Surfaces page



3. In the Terrain Patches list, select the terrain patch that you want to configure.
4. Select the check box for each soil type that you want to be visible.
5. Clear the check box for each soil type that you want to hide.
6. To change the state of multiple soil types at one time:
    a. Select (highlight) the names (not the check boxes) of the soil types whose state you want to change.
    b. Click Toggle Selected Surfaces.



65. Using Terrain Profiles

This chapter explains what terrain profiles are and how to display and configure them.

65.1. Terrain Profiles 1266
65.1.1 Displaying a Line's Terrain Profile 1267
65.1.2 Configuring Terrain Profiles 1267
65.2. Analyzing Terrain Using a Terrain Profile Line 1269

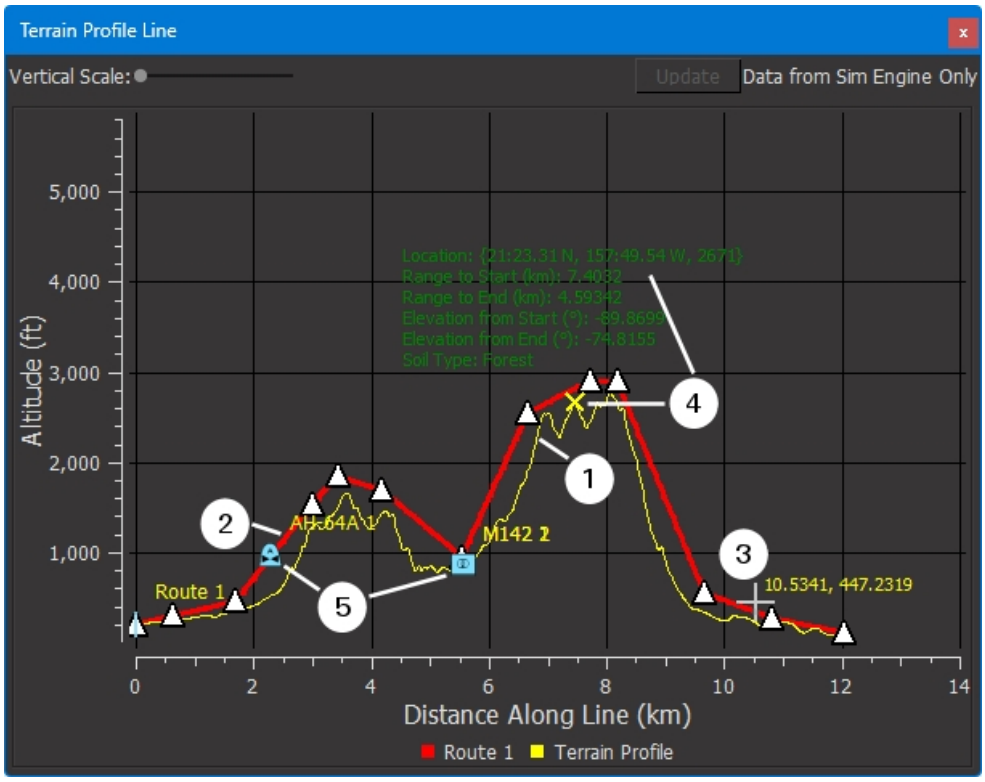
65.1. Terrain Profiles

You can display terrain profile windows for line objects that have height (in other words, routes and lines, but not phase lines), for intervisibility lines, and for a simulation object's track history. You can also create lines specifically for the purpose of showing the terrain profile. A terrain profile displays:

- A graph of the line against the terrain.
- Simulation objects within a specified distance of the line or track.
- Location of any one point along the line.
- The location on the grid of the cursor. This helps you estimate distances between the grid lines.

In Figure 517, the yellow line (1) represents the terrain and the red line (2) shows a route. If your cursor is in the grid, its location displays (3) as well as the location of one point along the terrain where we clicked (4). Several entities are close to the route and they are shown as well (5).

Figure 517. Terrain profile



65.1.1 Displaying a Line's Terrain Profile

You can display a terrain profile for routes, intervisibility lines, and terrain profile lines.

➤ **To view a line's terrain profile, use one of the following methods:**

- Right-click a line and choose **Terrain Profile** from the shortcut menu.
- Open the Edit dialog box for a line and click Terrain Profile. (You can configure terrain profiles to open automatically when you open an Edit dialog box.)

To display the terrain profile for a track history, follow the procedure in "50.2.4 Displaying a Terrain Profile for a Track History" on page 1057.

65.1.2 Configuring Terrain Profiles

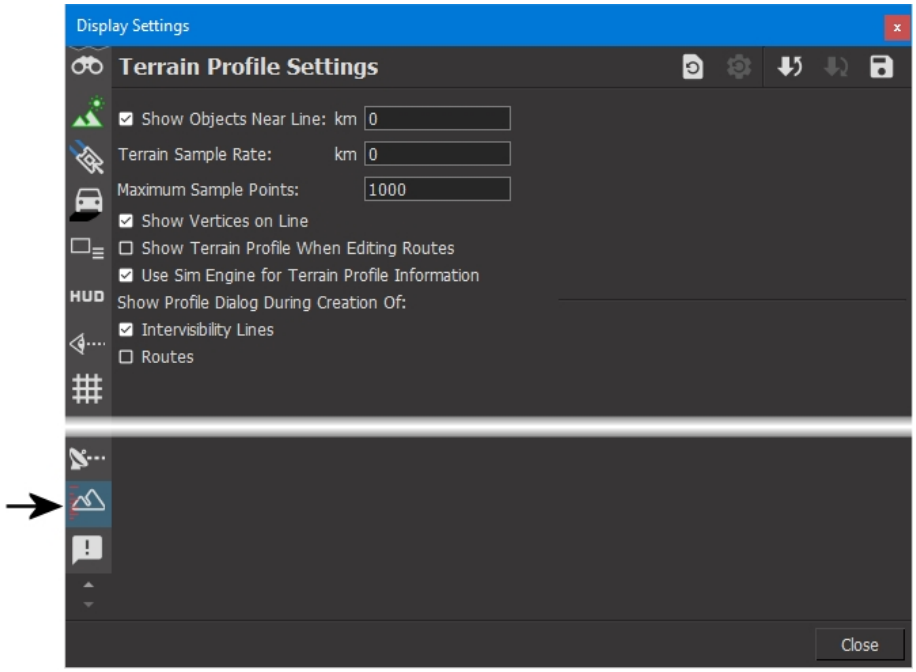
You can configure these aspects of a terrain profile:

- The frequency, in distance, at which terrain height is sampled to create the graph in the Terrain Profile window.
- Whether to use the sim engine for the terrain's height data when displaying the terrain profile line.
- The maximum number of points to sample along a line.
- Whether to show simulation objects that are close to the line and how close to the line simulation objects need to be to be shown.
- Whether to show the vertices of the line.
- Whether terrain profiles should be displayed automatically when a line is created or edited.

➤ **To configure terrain profiles:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Terrain Profile Settings page (Figure 518).

Figure 518. Terrain Profile Settings page



3. Change settings as desired. Table 53 describes the configuration options:

Table 53. Terrain profile parameters

Option	Description
Show Objects Near Line	If enabled, displays icons for simulation objects near the line. The text box specifies the distance from the line within which simulation objects are displayed.
Terrain Sample Rate	Specifies the frequency at which the terrain is sampled along the line.
Maximum Sample Points	Specifies the maximum number of sample points for a line.
Show Vertices On Line	If enabled, the vertices of the line are displayed.
Show Terrain Profile When Editing Routes	If enabled, a terrain profile window opens when you edit a route.

Table 53. Terrain profile parameters (continued)

Option	Description
Use Sim Engine for Terrain Profile Information	If enabled, the height data of the terrain for the Terrain Profile Line comes from the sim engine. While you create the line, the profile uses GUI data. When you finish creating the line, the height information comes from the sim engine. The Terrain Profile Line indicates whether the data is coming from the GUI or sim engine.
Show Profile During Creation	If enabled, a terrain profile window opens while intervisibility lines, routes, or both are created.

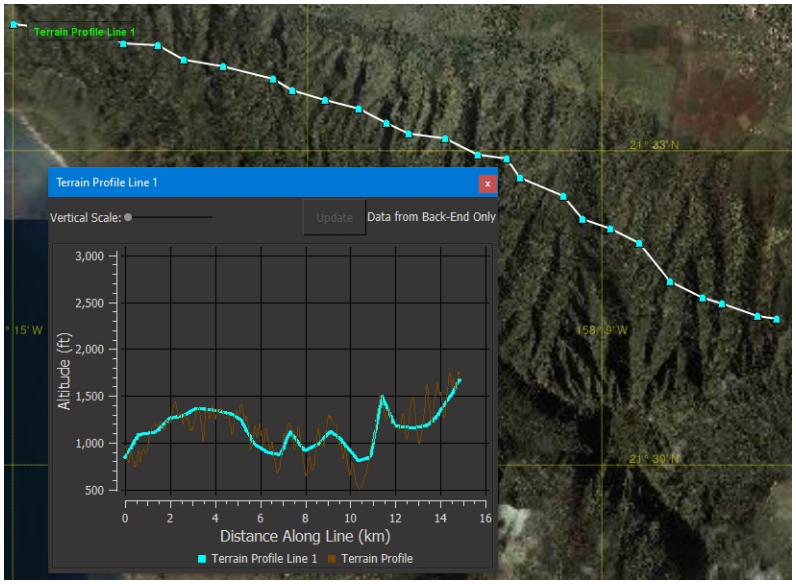
65.2. Analyzing Terrain Using a Terrain Profile Line

A terrain profile line is a special line used to analyze the terrain. It is more versatile than a point-to-point intervisibility line because it can have multiple points, which allows a finer analysis of the terrain. Unlike a route (for which you can also view terrain profiles), a terrain profile line is not published, so it has less overhead and does not clutter up object lists.

➤ **To create a terrain profile line:**

1. Choose **Intervisibility > Create Terrain Profile Line**. A Terrain Profile window opens and the cursor changes to draw mode.
2. Click on the terrain to place the vertices of the line. As you add vertices, the Terrain Profile window updates to show the profile of the line (Figure 519).

Figure 519. Terrain profile line



3. Double-click to finish the line.

After you create a terrain profile line, you can edit it just like any other line. For details, see "47.10.1 Editing Vertices" on page 1030.



66. Generating Navigation Data

This chapter explains how to generate navigation data for use with VR-Forces.

66.1. Introduction 1272
66.1.1 How Navigation Data is Generated 1272
66.1.2 Path Finding 1274
66.2. Navigation Areas 1276
66.2.1 Sectorizing Navigation Areas 1277
66.2.2 Creating a Navigation Area 1277
66.2.3 Editing a Navigation Area 1280
66.2.4 Reverting Edits to a Navigation Area 1280
66.2.5 Displaying Navigation Areas 1280
66.2.6 Removing a Navigation Area 1280
66.2.7 Displaying Information about a Navigation Area 1280
66.3. Generation of Navigation Data 1281
66.3.1 Navigation Profiles 1281
66.3.2 Generating Navigation Data for a Navigation Area 1282
66.3.3 Using Newly Generated Navigation Data 1284
66.3.4 Automatic Regeneration of Navigation Data 1284
66.4. Generating Navigation Data for Entities 1285
66.5. Importing Navigation Areas and Navigation Data 1287
66.6. Displaying Navigation Data 1287
66.6.1 Opening Navigation Lab from an Information Dialog Box 1288
66.6.2 Showing Simulation Object Movement in Navigation Lab 1289

66.1. Introduction

To use advanced navigation, you must use a terrain that has navigation data generated for the areas in which you want to navigate. You generate navigation data in the VR-Forces GUI.

When you generate navigation data, the VR-Forces creates separate sets of navigation data, at various levels of detail, for specified profiles. VR-Forces includes profiles for lifeforms and ground vehicles. You can modify the profiles and add new ones. The navigation data is stored in separate directories where it can be accessed by multiple terrains, if needed.

The general procedure for generating navigation data is:

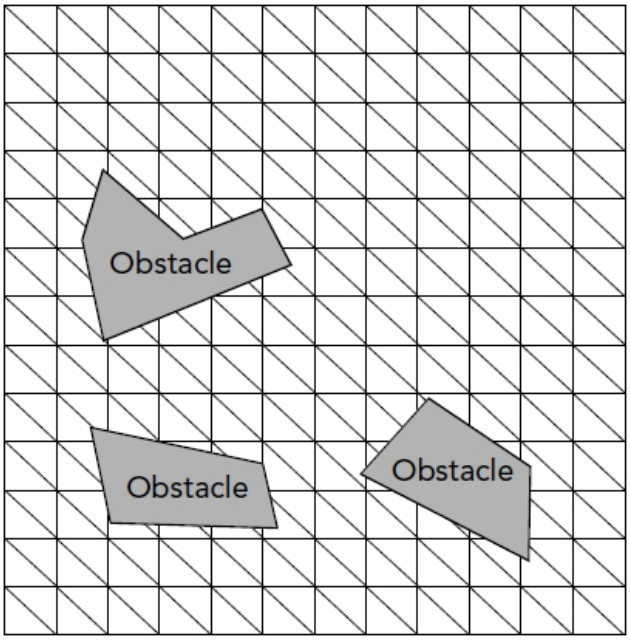
1. Load the terrain on which you want to generate navigation data or load a scenario that uses that terrain.
2. Create a navigation area.
3. Generate the navigation data.
4. Save the terrain.
5. Start a new scenario or save and reload the existing scenario.

66.1.1 How Navigation Data is Generated

Navigation data is generated from the GUI. Before you can use newly generated navigation data, you must close your scenario and reopen it. (This is so it can be loaded by the sim engine.) If the terrain is modified, the navigation data must be generated again.

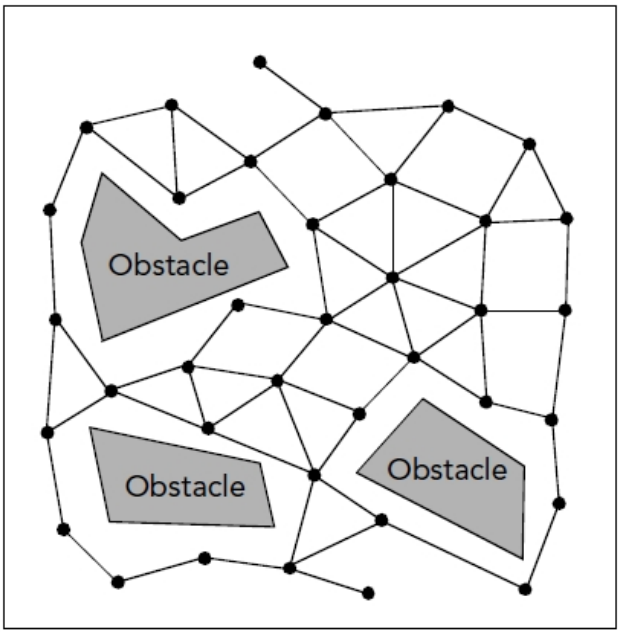
VR-Forces analyzes the terrain and creates a navigation mesh (NavMesh) of the terrain. The NavMesh defines areas in which an entity can move, given its dimensions. A lifeform can move in spaces that a ground vehicle cannot move in. Figure 520 shows a NavMesh for a terrain that has several areas in which entities cannot move.

Figure 520. NavMesh



Based on the NavMesh, VR-Forces generates a graph of locations in the terrain that the entity type can move to. Each location can be accessed from at least one other location. Figure 521 shows a hypothetical graph of the paths that an entity can follow in the NavMesh shown in Figure 520.

Figure 521. Graph



For details about generating navigation data, see "66.3. Generation of Navigation Data" on page 1281.

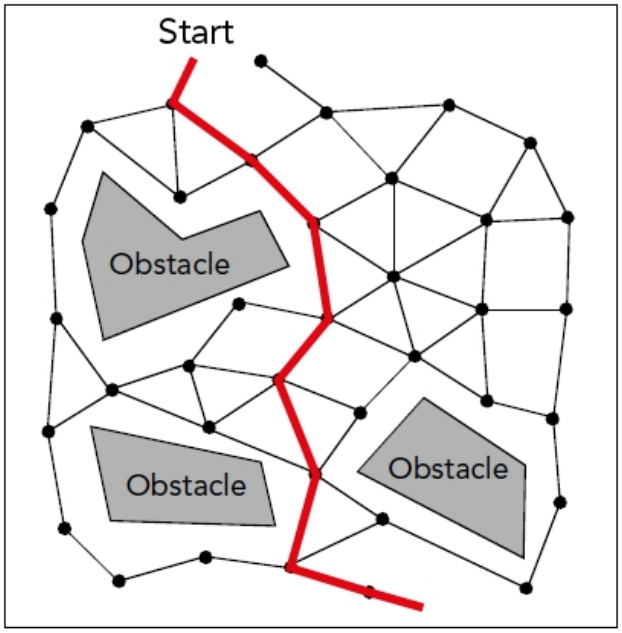
66.1.2 Path Finding

Path finding is the process by which an entity plans a path, follows the path, and avoids obstacles along the path.

Planning a Path

Every time an entity decides to move (either through an autonomous decision or a task assignment), it computes its path. Path computation, or path planning, consists of running an A* algorithm, with a given heuristic, on the navigation data.

Figure 522. Planning a path



A path finding heuristic is a cost estimate for the shortest path to reach a target. This cost can be measured according to various criteria, such as distance, time, and danger. The path finding heuristic is critical for the performance of the A* algorithm used by VR-Forces.

The standard path finding heuristic is the Euclidian Heuristic. In this heuristic, the cost of moving from one vertex to another vertex is the Euclidian length of the connecting edge.

Path Following

Once a path is determined for an entity, it can start moving to its final goal. As illustrated in Figure 522, a path is a set of vertices and edges. However, entities do not necessarily follow the exact path. VR-Forces smooths the entity's movement.

The first intermediary destination (or subgoal) of the entity is the nearest vertex. Before moving towards this point, the entity checks to see if it can move directly to the next vertex in its path. This test is performed against the AI mesh. If the next vertex is reachable, it becomes the new subgoal. The entity continues testing successive vertices until it finds a vertex that it cannot reach directly. The final trajectory is a smooth line that connects the various subgoals.

Figure 523 and Figure 524 illustrate this process. Using the prospective path defined in Figure 522, the entity computes its actual path. In Figure 523, view 1, it tests the first three vertices (dashed lines) and finds that it can reach all of them directly. In Figure 523, view 2, it tests the fourth vertex and finds that it can reach it. It then tests the fifth vertex and finds it cannot reach it directly. Therefore, the fourth vertex becomes the subgoal for this portion of the path. Figure 524 shows the final calculated path (dashed line). Notice how the path to the fourth vertex is much smoother than a path that strictly follows the vertices and edges would be, because the entity determined that it could move directly to that vertex.

Figure 523. Checking next vertex

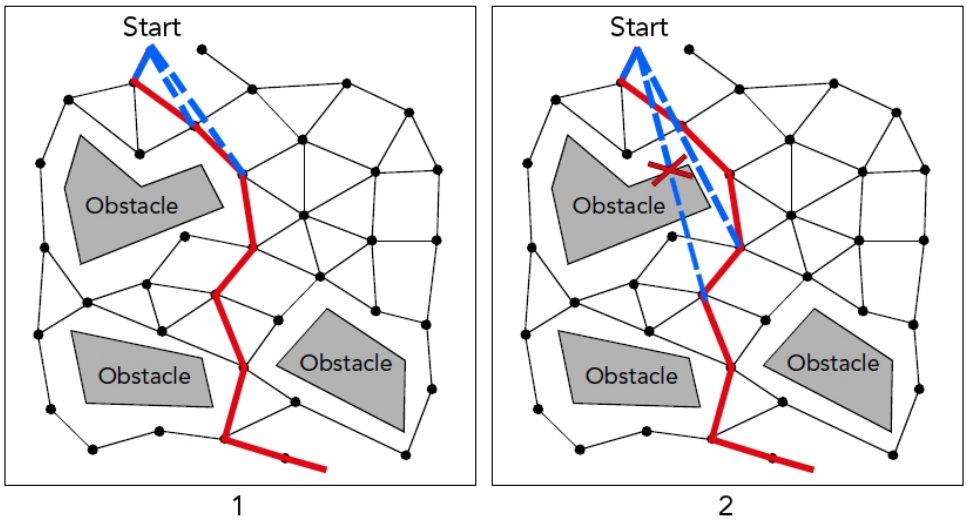
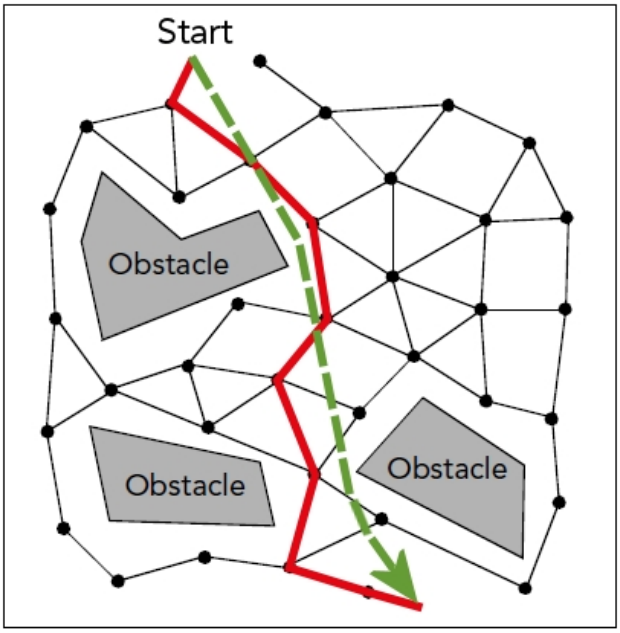


Figure 524. Final calculated path

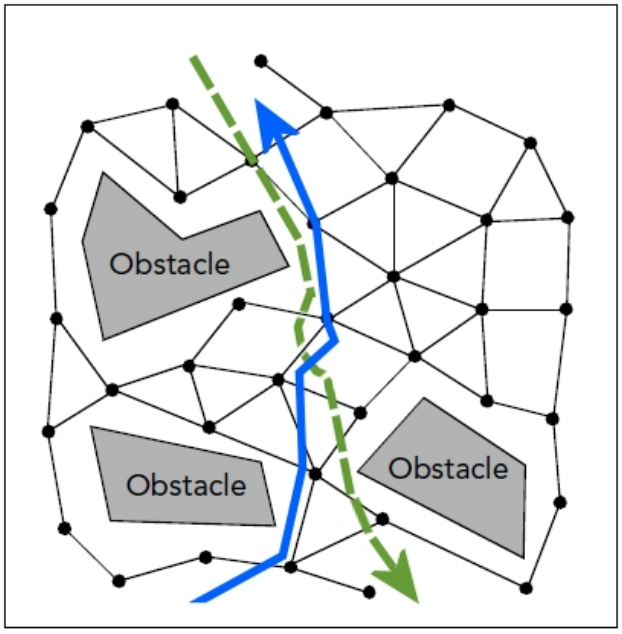


Dynamic Avoidance

While following its path, an entity may encounter other entities. VR-Forces models include collision avoidance. However in complex scenarios, the scenario developer may have to spend a considerable amount of time creating routes and coordinating entities to minimize collisions among entities and obstacles.

Advanced navigation uses a dynamic avoidance algorithm to make entities behave realistically when they are about to collide. Entities detect potential collisions in advance and adjust their trajectories smoothly using the navigation data.

Figure 525. Collision avoidance



In Figure 525, two entities traveling in opposite directions avoid each other, then resume their routes. Compare the track of the dashed line to that in Figure 524.

66.2. Navigation Areas

Navigation areas define areas on the terrain for which you can generate navigation data. Navigation areas are rectangular areas that are aligned to the North/South axis. The maximum size of the area on which you can generate navigation data depends on the raster precision value used to generate it. If you reduce the precision of navigation data, creating a sparser graph, the size of the maximum area increases. The maximum size for a navigation area using the default values is 20 km by 20 km. (Raster precision is configured in *./appData/settings/vrfSim/navigationProfiles.mtl*.)

You can create multiple navigation areas on a terrain.

66.2.1 Sectorizing Navigation Areas

You can generate navigation data for a navigation area as a whole or you can divide the navigation area into sectors and generate navigation data for each sector. VR-Forces can regenerate navigation data at runtime in response to changes in the terrain (dynamic terrain damage and placement of buildings). If you sectorize the navigation area, regeneration is much quicker because VR-Forces can restrict regeneration to the sector that is affected rather than the entire navigation area.

The disadvantages of using sectors are:

- Increasing the number of sectors slows down generation of navigation data.
- The navigation data generation process can fail if any single sector is too large, too complex, or both. However, this is also true if you generate navigation data for a very large terrain without using sectors.
- Path planning over long distances may be slower.

To generate a maximum size navigation area, you must use multiple sectors.

66.2.2 Creating a Navigation Area

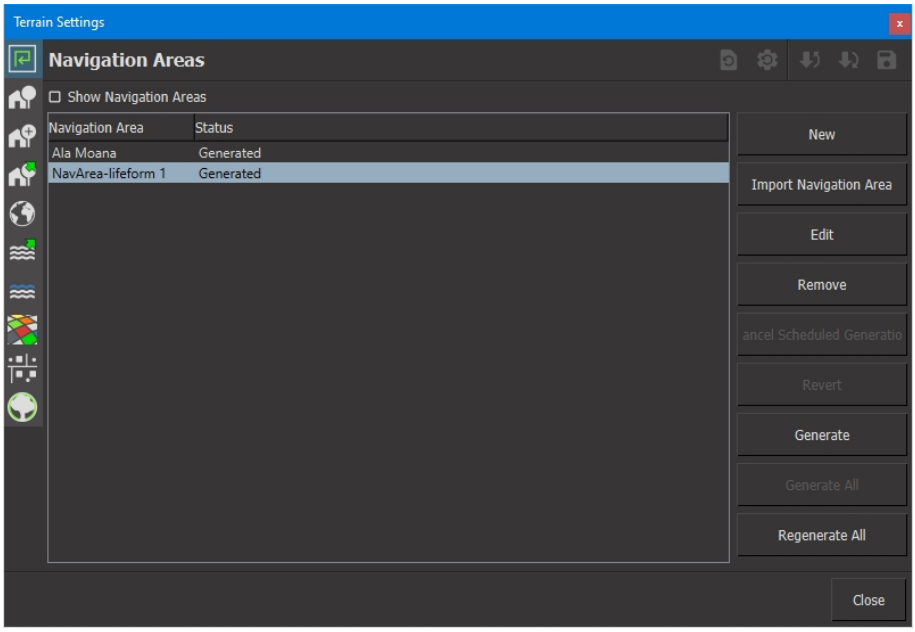
Navigation areas are created on the terrain used by the simulation engine. In most cases, scenarios use the same terrain for the GUI and simulation engine and there is no need to think about where navigation areas are created. However, sometimes scenarios use different terrains for the GUI and simulation engine. In those cases, note:

- The GUI only displays the navigation areas that are in the simulation terrain. If the GUI terrain has navigation areas they are not displayed.
- When you create a navigation area, it displays in the coordinate system used by the GUI terrain. Any data entry is in the GUI's coordinate system. However, the navigation area configuration files are saved in the coordinate system of the simulation terrain.
- The simulation terrain must be available to the GUI. If the GUI and simulation engine are running on different machines, it is possible that the files for the terrain used by the simulation engine are not on the computer that is running the GUI. However, for navigation areas to be created, the GUI must have access to the simulation terrain.
- Props are created on the GUI terrain. If you want props to be part of the navigation data, you must add them in a scenario that has the same terrain for the GUI and simulation engine.

➤ **To create a navigation area:**

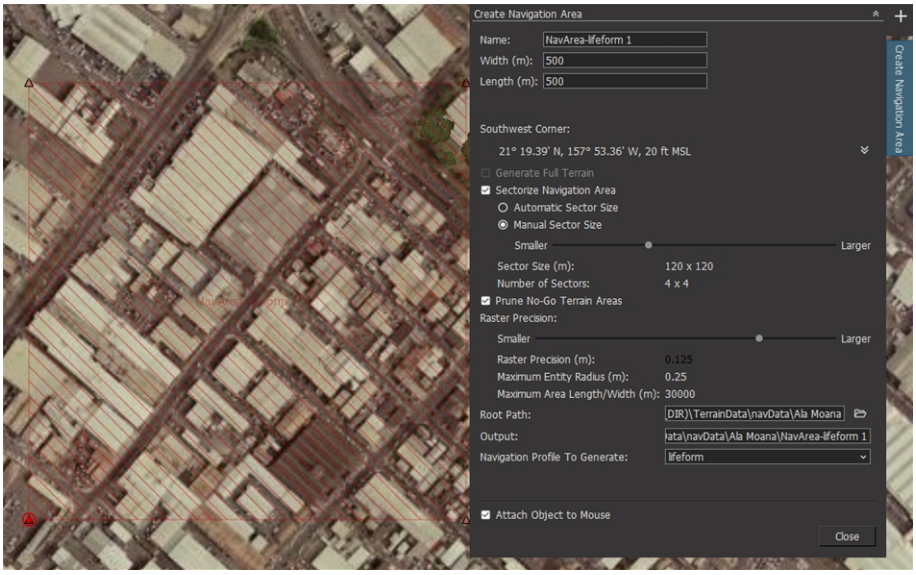
1. Open a scenario or a terrain on which you want to generate navigation data.
2. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 526).

Figure 526. Navigation Areas page



4. Click New. The cursor changes to create mode and shows a box that represents the navigation area. The default size is 500 m by 500 m. The Create Navigation Area tab is added to the window.
5. Click to place the lower left corner of the navigation area (Figure 527).

Figure 527. New navigation area



6. In the Create Navigation Area dialog box, edit the properties of the navigation area. Table 54 describes the fields in the dialog box.

Table 54. Navigation area properties

Property	Description
Name	The name of the navigation area. You cannot change the name after you create the navigation area.
Width	The length and width of the navigation area.
Length	Do not try to create an area larger than 20 km by 20 km unless you have changed the raster precision to support a larger area.
Southwest Corner	The coordinates of the southwest corner of the navigation area. (The altitude value is not used.)
Generate Full Terrain	Generate navigation data for the entire terrain. If the terrain is too large for this option, it is disabled.
Sectorize Navigation Area	Generate navigation data for sectors, rather than one large area.
⚠ If you are creating a large navigation area, we recommend that you always sectorize it. Generation of large unsectorized navigation areas sometimes fails.	
Automatic Sector Size	If selected, VR-Forces chooses a sector size based on the size of the navigation area. The sector size and number of sectors is shown below the manual sector size slider.
Manual Sector Size	Allows you to choose a sector size using the slider. As you move the slider, the sector size and number of sectors is shown below the slider.
Prune No-Go Terrain Areas	Do not calculate navigation data for areas in which simulation objects cannot move.
Raster Precision	Determines the density of the navigation graph.
Root Path	The root path of the data for this terrain.
Output (Path)	The location to save navigation data to.
Navigation Profile to Generate	Select the platform type for which you want to generate navigation data.
ⓘ Navigation areas created before VR-Forces 4.8 allowed more than one navigation profile. This is supported for backward compatibility only. For faster navigation generation, we recommend one profile per navigation area.	

7. Click Create.

66.2.3 Editing a Navigation Area

After you create a navigation area, you can edit any of its attributes except its name.

➤ **To edit a navigation area:**

1. Open a scenario or a terrain that has the navigation area you want to edit.
2. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 526).
4. Select the navigation area you want to edit.
5. Click Edit. The Edit Navigation Area dialog box opens.
6. Make the change you want to make.
7. Click OK.

66.2.4 Reverting Edits to a Navigation Area

If you edit a navigation area, you can revert to the previous version of the area.

➤ **To revert an edited navigation area:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 526).
3. Select the navigation area you want to revert.
4. Click Revert.

66.2.5 Displaying Navigation Areas

➤ To display navigation areas, choose **Settings > Navigation Areas**, click the Navigation Area button on the Display Settings toolbar, or select the Show Navigation Areas check box on the Terrain Settings dialog box, Navigation Areas page.

66.2.6 Removing a Navigation Area

➤ **To remove a navigation area:**

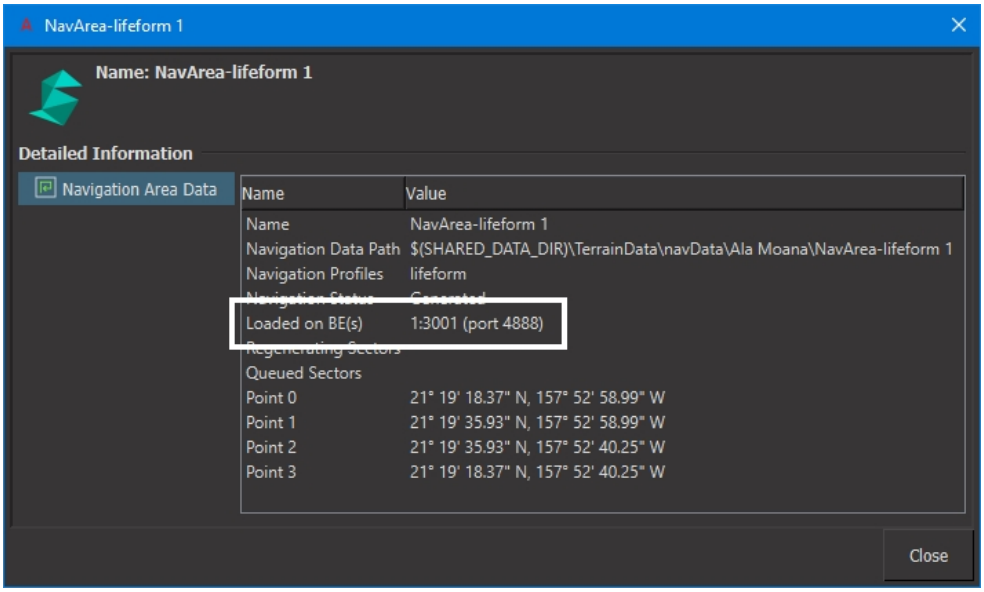
1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 526).
3. Select the navigation area you want to remove.
4. Click Remove.

66.2.7 Displaying Information about a Navigation Area

You can display an Information dialog box that has information about the selected navigation area (Figure 528). By default, the dialog box does not display the port used by the navigation area. It is marked (Disabled). To display the port, in *./appData/settings/vrfSim/vrfSim.mtl*, set **enableNavigationLabDebugging** to 1.

For performance reasons, navigation data does not load until you place a simulation object that will use it. When you enable Navigation Lab debugging and look at the terrain's navigation areas before a simulation object uses it, the information dialog box shows that the data has not been loaded.

Figure 528. Navigation area information



66.3. Generation of Navigation Data

To generate navigation data, you create navigation areas and then generate navigation data for them. A terrain can have more than one navigation area. Entities use advanced navigation within the areas and standard navigation if they move between them.

Generating navigation data can last from a few minutes to hours to days depending on the terrain complexity, the level of detail, and your computer's processing power. Navigation data is saved to *./userData/navData* in a directory that matches the name of the terrain on which it was generated. In addition to using sectors ("66.2.1 Sectorizing Navigation Areas" on page 1277), generating cover points can significantly affect the amount of time it takes to generate navigation data.

By default, when you generate navigation data for lifeforms, VR-Forces calculates cover points. For large and complex terrains, this process can be very slow. MAK has recorded this taking one week to finish on a maximum size navigation area. You can disable calculation of cover points in the navigation profile. However, if you disable calculation of cover points, tasks that use cover points will not work in the navigation area.

66.3.1 Navigation Profiles

VR-Forces generates navigation data based on parameters in a profile. VR-Forces includes profiles for lifeforms and ground entities. They provide default values for generating navigation data that should work for most terrains. You can change the profiles. You can also add profiles. Profiles are configured in *./appData/settings/vrfSim/navigationProfiles.mtl*. The configuration parameters are explained in the file.

Navigation profiles include parameters that control the priority of navigation data regeneration in response to terrain changes and the types of changes that trigger regeneration. For more information about regenerating navigation data, see "66.3.4 Automatic Regeneration of Navigation Data" on page 1284.

You can use the `nav-edges` block in a navigation profile to connect points in the terrain that would not normally be connected by the navigation data generation process. This feature, for example, allows a human entity to walk up to a door, open the door, walk through the door, and then close the door. This is an advanced feature. For details, see the comments in *navigationProfiles.mtl* or contact support at https://www.mak.com/mak-one/support/portal.

> You can dynamically add nav edges to a navigation area using nav edge tactical graphics. For details, see "47.6. Creating Nav Edges" on page 1023.

The profile that an entity uses for advanced navigation is specified in its OPE file. For example, *Human.ope* specifies use of the lifeform profile:

```
(navigation-parameters
  (DtRwString navigation-profile  "lifeform")
  (DtRwString navigation-method  "shortcut")
)
```

Another advanced feature that you can use is `min-navigable-surface`, which indicates the minimum size, in meters squared, of the surface area that a section of the terrain must have before the generator will generate navigation data for it. The default is 10 meters squared, which is reasonable for many terrains but can omit small rooms that are connected by dynamic doors to the larger terrain. By specifying smaller sizes, you can include areas such as those.

> If required for the terrain you will use in your simulation, you can specify which soil types are tagged with surface characteristics. There is a configuration option in *navigationProfiles.mtl*, `soil-types-to-tag-with-surface-char`, which lists soil types. Only soil types included in the list will generate navigation tags specific to that soil type. Other soil types receive the default navigation tag. For details, see that section of the file.

66.3.2 Generating Navigation Data for a Navigation Area

The Navigation Areas page (Figure 526) displays the status for each navigation area:

- Generated. Data has been generated.
- Needs Generation. The navigation area needs to have navigation data generated.
- Scheduled Generation. VR-Forces is generating navigation data for multiple navigation areas. This area is in the queue for generation.

You can generate navigation data for:

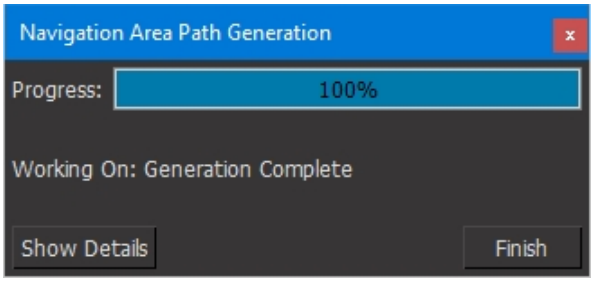
- Selected navigation areas.
- For all navigation areas that need to have navigation data generated.
- For all navigation areas, including those that already have navigation data.

When you generate navigation data, the generator does not use the version of the terrain that is loaded into the simulation engine or GUI. It uses a separate version. Therefore, if you have made any changes to the terrain, for example by adding props, and you want them to be part of the updated terrain, you must save the terrain before you generate navigation data.

➤ **To generate navigation data:**

1. Open a scenario or a terrain on which you want to generate navigation data.
2. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
3. Select the Navigation Areas page (Figure 526 on page 1278).
4. Select the navigation areas for which you want to generate navigation data.
5. Click Generate.
6. You are prompted to save the terrain before navigation data is generated.
7. Click Yes. The Navigation Area Path Generation dialog box opens (Figure 529). You can optionally view details of the generation process or cancel the process. When the navigation data is generated, the dialog box buttons redisplay. You are prompted to reload the scenario.

Figure 529. Navigation Area Path Generation dialog box



8. Click OK.
9. Click Finish.
10. Save the terrain. (**File > Terrain > Save Terrain**.)
11. To use the navigation data, reload the scenario.

Removing a Navigation Area from the Generation Queue

If you schedule multiple navigation areas for navigation data generation by clicking Generate All or Regenerate All, you can remove areas from the generation queue. If you

remove a navigation area for which you had already generated navigation data, its status returns to Generated. If you remove a navigation area that has not had navigation data generated, its status returns to Need Generation.

➤ **To remove a navigation area from the generation queue:**

1. Select the navigation area in the Navigation Areas list.
2. Click Cancel Scheduled Generation.

Canceling Generation of Navigation Data

You can cancel generation of navigation data while it is being generated for a navigation area. If you cancel generation of navigation data, the status for the navigation area changes to Needs Generation, even if you previously generated navigation data for this area.

➤ **To cancel generation of navigation data while it is being generated, do one of the following:**

- Select the navigation area in the Navigation Areas window and click Cancel Scheduled Generation.
- In the Navigation Area Path Generation dialog box, click Cancel NavArea *n*, where *n* is the number of the area currently being generated.

66.3.3 Using Newly Generated Navigation Data

To use newly generated navigation data, create a scenario that uses the terrain or open a scenario that uses that terrain. If you generate navigation data while a scenario is open, you must close the scenario, then open it again before you can use the navigation data.

66.3.4 Automatic Regeneration of Navigation Data

By default, VR-Forces automatically generates navigation data if a dynamic terrain change occurs in a navigation area. Unless your navigation area is quite small, regenerating navigation data will be much quicker if you create the navigation area with sectors.

The speed at which navigation data is regenerated depends on different factors (most notably sector size, number of affected sectors, and the complexity of the terrain), but in all cases it is not instantaneous. It takes at least several seconds, and possibly considerably longer. It is not abnormal for regeneration to take up to 20-30 seconds even in a well sectorized terrain. Therefore it is normal for entities to continue navigating using the original navigation data for a brief period until navigation data is updated.

Regeneration of navigation data is configured in *./appData/settings/vrfSim/navigationProfiles.mtl*.

You can specify the types of dynamic terrain change that trigger regeneration of navigation data and a priority for how quickly regeneration takes place relative to other profiles. The following code configures regeneration of navigation data for the lifeform profile. Please see the file for explanatory comments.

```
(regeneration-priority 1)
```

```
(regenerate-for-dynamic-terrain-types
  (dynamic-terrain-type  "entity")
  (dynamic-terrain-type  "damage")
  (dynamic-terrain-type  "damage_structure")
)
```

Regeneration of navigation data in response to terrain changes does not require an additional license. Therefore, terrains that have navigation areas (which require an extra cost license to generate) can be distributed to VR-Forces users and regeneration of navigation data will work even if those users do not have a license for creating navigation areas.

66.4. Generating Navigation Data for Entities

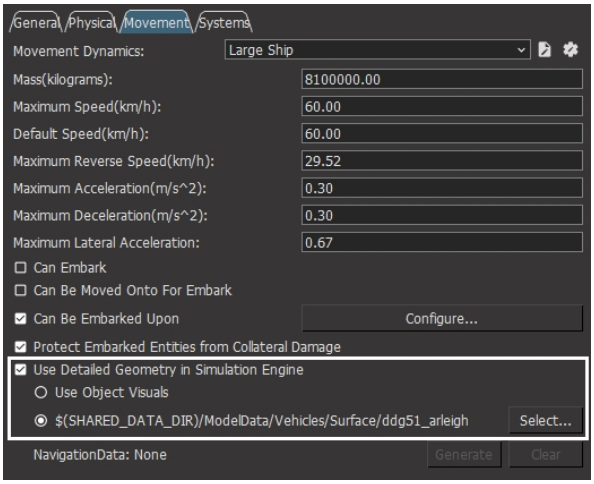
You can generate navigation data for entities that have object geometry. This allows human and ground entities to move around on the deck of an aircraft carrier or some other surface entity. Navigation data is generated for any surface on the entity that is accessible for the profile. For example, lifeforms can move on steeper surfaces than ground platforms. You generate navigation data for entities in the Simulation Object Editor. For complete details about using the Simulation Object Editor, see "Section XIII. Creating and Editing Simulation Models" on page 1301.

When you generate navigation data for an entity, it is saved in the *vrfSim/navData* directory for the SMS (*./data/simulationModelSets/<model_set>*). It saves immediately, regardless of whether you save changes to the SMS.

➤ **To generate navigation data for an entity:**

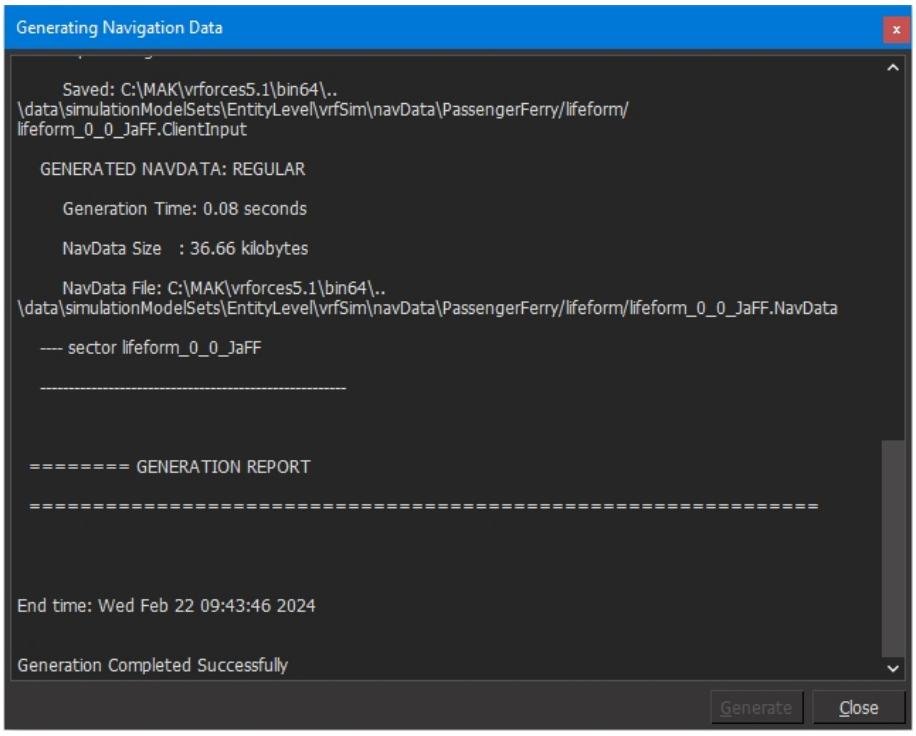
1. Start the Simulation Object Editor, as described in "68.2. Starting the Simulation Object Editor" on page 1305.
2. Load the SMS for the entity you want to edit, for example *EntityLevel.sms*.
3. In the object list, select the entity for which you want to generate navigation data.
4. Select the Movement tab.
5. Select the Use Detailed Geometry in Simulation Engine check box (Figure 530).

Figure 530. Object geometry



6. If you have not specified geometry for this entity, click Select and select the appropriate geometry file. (For complete details, see "70.1.6 Adding Object Geometry to a Simulation Object" on page 1380.)
7. Click Generate. The Generating Navigation Data dialog box opens.
8. Specify the platform types for which you want to generate navigation data.
9. Click Generate. VR-Forces generates navigation data (Figure 531). When the process finishes, the Cancel button changes to read Close.

Figure 531. Generating Navigation Data dialog box



10. Click Close.

66.5. Importing Navigation Areas and Navigation Data

When you generate navigation data for a navigation area, navigation area configuration files and the navigation data itself are saved to *./userData/navData*. For example, the configuration files for the Ala Moana navigation area provided with VR-Forces are *Ala Moana.navGenConfig* and *Ala Moana.navRuntimeConfig*.

You can import the navigation area and navigation data into other terrains that have the same coordinate systems and cover the same area of the earth. When you import a navigation area, it is imported with the status Generated.

➤ **To import a navigation area and its associated navigation data:**

1. Choose **Settings > Terrain**. The Terrain Settings dialog box opens.
2. Select the Navigation Areas page (Figure 526 on page 1278).
3. Click Import Navigation Area. A file browser window opens.
4. Navigate to the directory for the navigation area that you want to import and select the *.navRuntimeConfig* configuration file for the navigation area.
5. Click Open. The navigation area is added to the terrain.
6. Save the terrain.

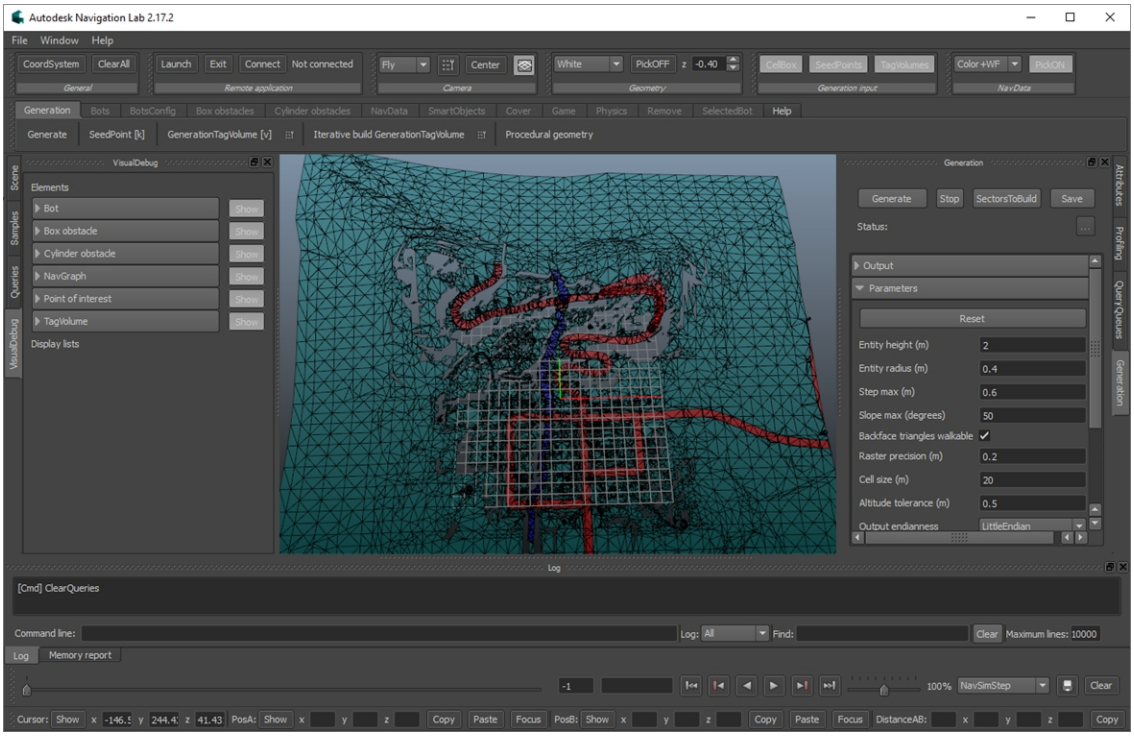
66.6. Displaying Navigation Data

VR-Forces includes Autodesk® Gameware Navigation Lab software (Navigation Lab), a tool that lets you view navigation data. It also lets you connect to a simulation and watch entities move about on the terrain.

You can view navigation data for navigation areas and for the object geometry on entities. You can open Navigation Lab and use its menus to select the navigation data to view or you can open it directly from an Information dialog box.

Figure 532 displays the navigation data for VR-Village.

Figure 532. Navigation data display



Autodesk® Gameware Navigation Lab is not supported on Linux.

➤ **To display navigation data by opening it in Navigation Lab:**

1. On the Start menu, choose **MAK Technologies > VR-Forces 5.2 Tools**. In the Tools folder, double-click the Navigation Lab shortcut. Navigation Lab opens.
2. Choose **File > Open**. A file browser opens.
3. To view navigation data for a terrain, navigate to a terrain under *./userData/navData*. To view navigation data for an entity, navigate to a directory in *./data/simulationModelSets/EntityLevel/vrfSim/navData*.
4. Select a file with the *.NavData* extension. The navigation data is displayed in the Navigation Lab window.

66.6.1 Opening Navigation Lab from an Information Dialog Box

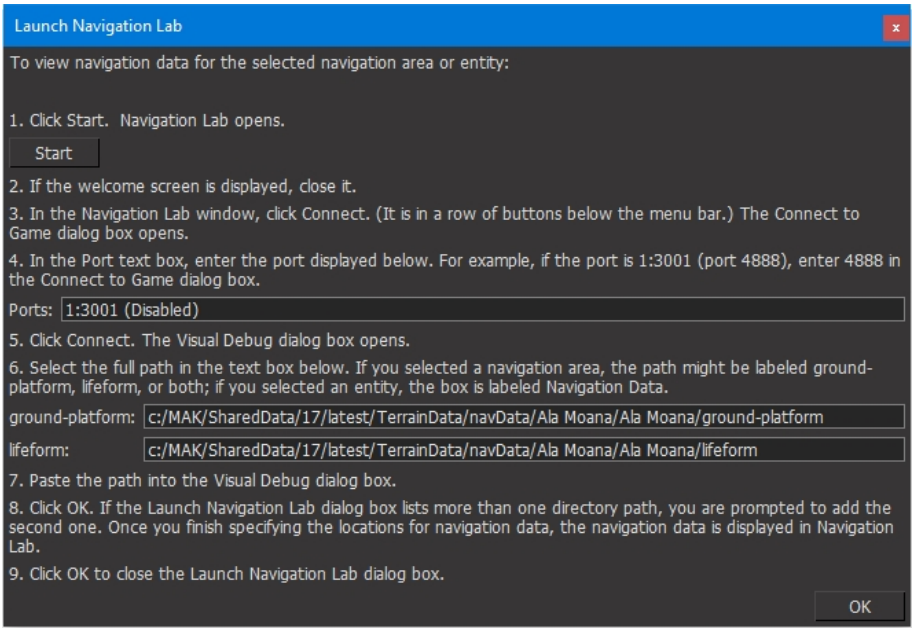
Navigation areas and entities that have object geometry and navigation data have a button on their Information dialog box that you can use to quickly display their navigation data in Navigation Lab.

Navigation data does not load until it is needed, such as when an entity is placed in the area. Until navigation data is loaded, clicking the Navigation Lab button displays a message informing you that you can use the Navigation Lab once the navigation data is loaded in the sim engine.

➤ **To open Navigation Lab from an Information dialog box:**

1. Open the Information dialog box for the navigation area or entity for which you want to view navigation data.
2. Click the Navigation Lab button ( ). The Launch Navigation Lab dialog box opens (Figure 533).

Figure 533. Launch Navigation Lab dialog box



3. Follow the directions on the dialog box.

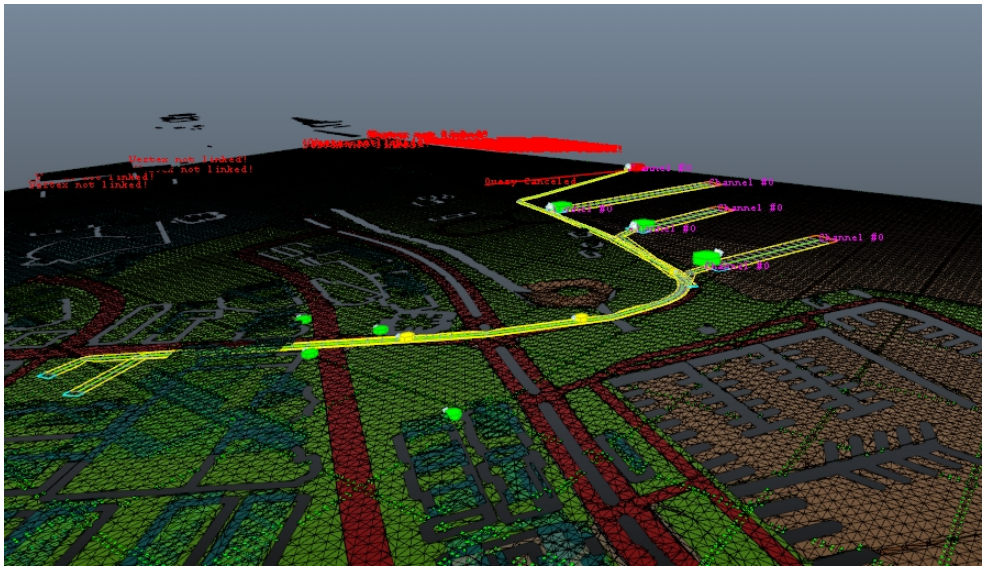
66.6.2 Showing Simulation Object Movement in Navigation Lab

You can view simulation objects in Navigation Lab. This can help you analyze problems in navigation data, such as the inability of an entity to plan a path.

➤ **To view simulation objects in Navigation Lab:**

1. Connect to Navigation Lab as described in "66.6.1 Opening Navigation Lab from an Information Dialog Box" on the previous page.
2. Run the scenario.
3. In Navigation Lab, look for volumetric objects that represent the simulation objects (Figure 534).

Figure 534. Simulation objects represented in Navigation Lab





67. Aerodromes

Aerodromes allow you to give aircraft realistic places to take off and land. You can also control managed aerodromes' settings either manually or as part of a plan. This chapter describes how to work with aerodromes that are in a terrain and their functionality.

67.1. Overview of Aerodromes 1292
67.1.1 Checking a Terrain for Aerodromes 1292
67.2. Managed Aerodromes and Runways 1292
67.2.1 Promoting a Specific Aerodrome 1292
67.2.2 Promoting Aerodromes in a Terrain 1293
67.2.3 Viewing Aerodrome Information 1294
67.2.4 Centering on an Aerodrome 1294
67.3. Aerodrome and Runway Set Data Requests 1295
67.3.1 Aerodrome Light Conditions 1295
67.3.2 Aerodrome Undercarriage Inspection System 1295
67.3.3 Lights (Advanced) 1295
67.3.4 Runway Active 1297
67.3.5 Runway Arrestor Gear 1297
67.3.6 Runway Condition 1298
67.3.7 Runway Damaged 1298
67.3.8 Runway Light Conditions 1298

67.1. Overview of Aerodromes

Aerodromes are representations of airports and airbases using data is contained in a shape file. Some of the terrains in the MAK data package include aerodrome data, and you can also use your own terrains if they have aerodrome data.

When aerodromes simply exist in the terrain, you can have aircraft take off and land from aerodromes' runways. You can also have aircraft take off and land on route tactical graphics instead of using aerodromes.

If you want to manage an aerodrome's features, such as lighting and runway conditions, you must promote it. You can select specific aerodromes to promote individually, or you can promote aerodromes in an area. After promoting aerodromes, you can manage them as needed for your scenario. You can control the aerodrome lights as well as the runway lighting and conditions, either as a set data request or part of a plan. The changes you make are reflected in any simulation connected to the exercise.

For entities to take off from and land on runways at managed aerodromes, the runways must be set to active.

> The aerodromes included in MAK terrains do not include helipads or water landing sites. They are primarily intended for fixed-wing aircraft that take off from and land on runways.

Aerodromes are a layer in MAK Earth and streaming terrains. The features are part of the vector files that are distributed with VR-Forces. These files are originally from X-Plane and, while they are good for many of the airports in the world, they may not be entirely accurate. You can use GIS tools to edit the files and change, add, or remove aerodrome features as needed.

67.1.1 Checking a Terrain for Aerodromes

It's simple to check whether a terrain includes aerodromes. After loading a terrain in VR-Forces, check the list in the Aerodromes toolbar. If there are values in it, then those aerodromes have been defined in the terrain. If the list is empty, then there are no aerodromes.

➤ To open the Aerodromes toolbar, choose **View > Toolbars > Aerodromes Toolbar**.

67.2. Managed Aerodromes and Runways

You can use either the Create Selected Aerodrome or Promote Aerodrome feature in terrains that include aerodrome shape files to promote aerodromes and their runways to be managed. This allows you to use set data requests to control the lighting and runway conditions.

67.2.1 Promoting a Specific Aerodrome

You can quickly promote an aerodrome from the Aerodromes Toolbar. This makes it a managed aerodrome in the neutral force, and one runway is set to Active.

➤ **To promote a specific aerodrome:**

1. If the Aerodromes Toolbar is not visible, choose **View > Toolbars > Aerodromes Toolbar**.
2. In the Aerodrome list, select the aerodrome to generate. You can filter the list by entering part of the name or code.
3. Click Create Selected Aerodrome ().

67.2.2 Promoting Aerodromes in a Terrain

Using Promote Aerodrome, you can promote a specific aerodrome, or some or all of the aerodromes that exist in an area of the terrain. You also specify the force for the aerodromes. When you promote aerodromes, VR-Forces uses the terrain data to get the information about the aerodrome and runway objects. One runway is also set to Active for the aerodromes when you promote them.

➤ **To promote a specific aerodrome:**

1. Choose **Create > Promote Aerodrome**. The Promote Aerodrome dialog box opens.
2. Select the force for the aerodrome.
3. Select the ICAO Code option.
4. In the Aerodrome list, select the aerodrome to generate. You can filter the list by entering part of the name or code.
5. Click Create. The aerodrome and its associated runways are created.

➤ **To promote some or all of the aerodromes in an area:**

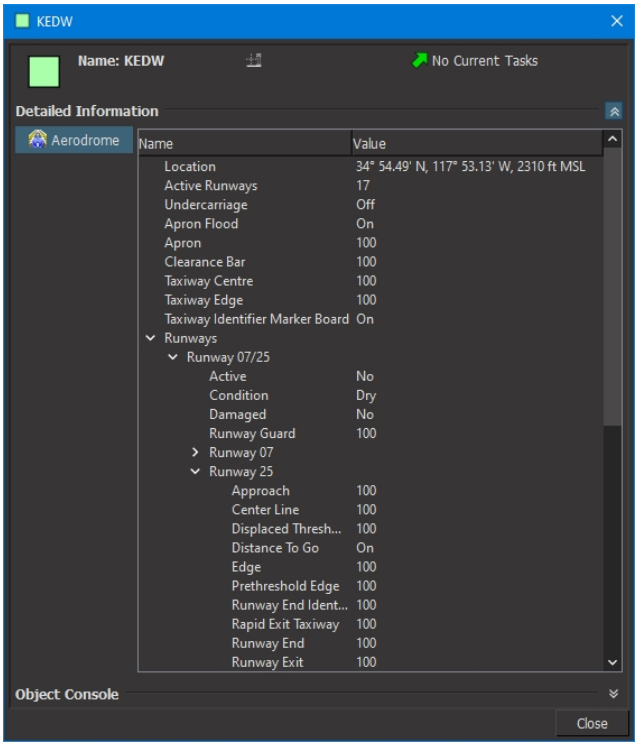
1. Adjust your view of the terrain in the main window to include the aerodromes you want to promote.
2. Choose **Create > Promote Aerodrome**. The Promote Aerodrome dialog box opens.
3. Select the force for the aerodromes.
4. Select the Area Selection option.
5. Click the Select button. The cursor changes.
6. On the terrain where the aerodromes are located, draw an area as you would any areal tactical graphic.

   When you finish the area, you return to the Promote Aerodrome dialog box and the Aerodromes list populates with the aerodromes defined in that area of the terrain. Depending on the number of aerodromes in the area, this can take a little while.
7. If desired, select or unselect aerodromes to limit the list to only the aerodromes you want to promote. You can use **Ctrl**+click or **Shift**+click.
8. Click Promote. The selected aerodromes are added to the Objects List as managed aerodromes. This can also take some time if you selected many aerodromes.

67.2.3 Viewing Aerodrome Information

The Information dialog box for managed aerodromes includes information on the lighting and conditions for the aerodrome and its runways.

Figure 535. Information dialog box for Aerodrome



67.2.4 Centering on an Aerodrome

There are a number of ways that you can center on an aerodrome.

➤ To center on a managed aerodrome from the Last Selected Object Panel or the Information dialog box, click Center on Aerodrome ( ).

➤ **To center on an aerodrome from the Aerodromes Toolbar:**

1. If the Aerodromes Toolbar is not visible, choose **View > Toolbars > Aerodromes Toolbar**.
2. In the Aerodrome list, select the aerodrome to center on. You can filter the list by entering part of the name or ICAO code.
3. Click Center on Aerodrome ( ). Your window focuses on the selected aerodrome.

As with other simulation objects, you can also double-click on an aerodrome in a list to center on it.

67.3. Aerodrome and Runway Set Data Requests

You can control the settings for managed aerodromes and their runways using set data requests, including in global or individual plans.

> Not all aerodromes in the terrains included with VR-Forces are configured for all supported lights. The Edwards Air Force Base (KEDW) has been fully configured as an example. Other aerodromes include some lights, but require terrain file changes to display others. See the *Procedural Airports* chapter in the *Adding Content to MAK ONE Applications Reference Manual* for more information.

67.3.1 Aerodrome Light Conditions

You can set the light conditions for all of the lights at an aerodrome, including the runway lights. The lighting changes are visible in Stealth observer mode (where configured), and your changes set the state on the managed objects and publish the information to the network. You can also see the lighting information in the aerodrome's Information dialog box.

➤ **To set the light conditions for an entire aerodrome:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Aerodrome Light Conditions**. The Aerodrome Light Conditions parameter displays.
3. From the list, select the desired light condition.
4. Click OK.

67.3.2 Aerodrome Undercarriage Inspection System

The Aerodrome Undercarriage Inspection System set data request allows you to set the undercarriage inspection system at an aerodrome to either on or off. There is no simulation effect associated with this setting. The information is only published to the network. You can also see the system's state in the aerodrome's Information dialog box.

➤ **To set the state of an aerodrome's undercarriage inspection system:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Aerodrome Undercarriage Inspection System**. The Undercarriage Inspection System parameter displays.
3. From the list, select whether the system is on or off.
4. Click OK.

67.3.3 Lights (Advanced)

Using the Lights (Advanced) set data request, you can set the intensity of the lights for a managed aerodrome as well as its associated runways. The lighting changes are visible in

Stealth observer mode, and your changes set the state on the managed objects and publish the information to the network. You can also see the light settings in the aerodrome's Information dialog box.

Lights you can control the state of include:

- Aerodrome
  - Apron
  - Apron Flood
  - Clearance Bar
  - Taxiway Centre
  - Taxiway Edge
  - Taxiway Identified Marker Board
- Runways
  - Approach
  - Center Line
  - Displaced Threshold
  - Distance To Go
  - Edge
  - Prethreshold Edge
  - Rapid Exit Taxiway
  - Runway End
  - Runway End Identifier
  - Runway Exit
  - Runway Guard
  - Taxiway Lead On
  - Taxiway Stop Bar
  - Threshold
  - Touch Down Zone
  - VSGI

You can use Edwards Air Force Base (KEDW) to visualize the lights. Other aerodromes can publish the information but might not be configured to display all of them.

➤ **To set the aerodrome or runway lights:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Lights (Advanced)**. The Aerodrome Lights parameters display.
3. Select the lights that you want to change. You can specify them for all of the lights in the aerodrome or all runways, or you can expand the list underneath any of them to choose specific lights.

4. Set the intensity using the slider or by entering a value in the box. The intensity is a percentage, from 0 (off) to 100%.
5. Click OK.

67.3.4 Runway Active

The Runway Active set data request allows you to specify whether the runway is active. When one direction of the runway is active, the other direction is inactive. There is no simulation effect associated with this setting. The information is only published to the network. You can also see the runway's active state in the associated aerodrome's Information dialog box.

➤ **To set the active state of runways:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Runway Active**. The Runway Active parameters display.
3. You can specify the active state for all of the runways, or you can expand the list underneath these to choose specific runways. For each runway or a collection of runways, specify the active state.
4. Click OK.

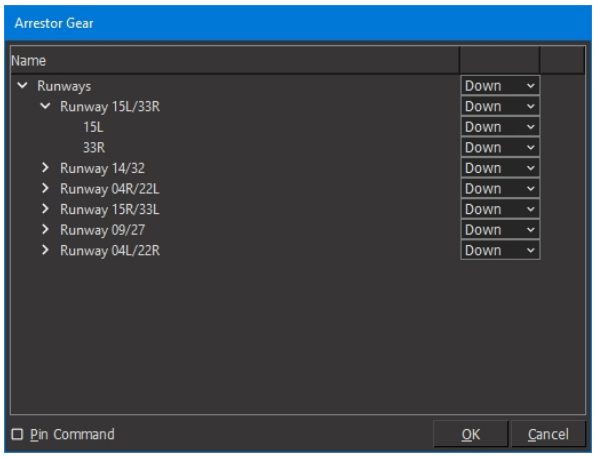
67.3.5 Runway Arrestor Gear

The Runway Arrestor Gear set data request allows you to set the arrestor gear on a runway to either up or down for one or more runways. There is no simulation effect associated with this setting. The information is only published to the network.

➤ **To set the state of runway arrestor gear at an aerodrome:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Runway Arrestor Gear**. The Arrestor Gear parameters display (Figure 536).

Figure 536. Arrestor Gear dialog box



3. You can specify the arrestor gear state for all of the runways, or you can expand the list underneath these to choose specific runways. For each runway or a collection of runways, specify whether the arrestor gear is up or down.
4. Click OK.

67.3.6 Runway Condition

The Runway Condition set data request allows you to specify the conditions on one or more runways at an aerodrome. The runway conditions are not visible in any views of the exercise, but your changes set the state on the managed objects and publish the information to the network. You can also see the runway condition in the associated aerodrome's Information dialog box.

If you set the runway condition to Snow, for example, there is no visualization of snow. If you want to see snow, you could add a local weather zone and set the desired environmental condition, such as Blowing Snow or Snowy. See "Chapter 11. Setting Environment Conditions" on page 323. Likewise, adding simulated weather does not affect a runway's condition setting.

➤ **To set runway conditions:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Runway Condition**. The Runway Surface Conditions parameters display.
3. Specify the runway surface condition for the desired runways.
4. Click OK.

67.3.7 Runway Damaged

The Runway Damaged set data request allows you to specify whether one or more runways at an aerodrome are damaged. The damage is not visible in any views of the exercise, but your changes set the state on the managed objects and publish the information to the network. You can also see the state of the runway in the associated aerodrome's Information dialog box.

➤ **To set runway conditions:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Runway Damaged**. The Runway Damaged parameters display.
3. Specify the damaged state for the desired runways.
4. Click OK.

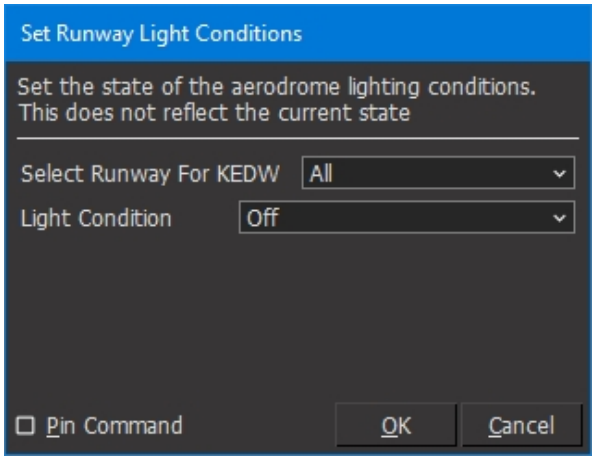
67.3.8 Runway Light Conditions

The Runway Light Conditions set data request allows you to set the lights for one or all runways at the aerodrome. The lighting changes are visible in Stealth observer mode, and your changes set the state on the managed objects and publish the information to the network. You can also see the light settings in the associated aerodrome's Information dialog box.

➤ **To set the light conditions for runways:**

1. In the Objects List, select a managed aerodrome.
2. Choose **Set > Runway Light Conditions**. The Runway Light Conditions parameters display (Figure 537).

Figure 537. Runway Light Conditions dialog box



3. From the Select Runway list, choose whether you want to set the light conditions for all of the runways at the aerodrome or select a specific runway.
4. Select the desired light condition from the list.
5. Click OK.



XIII. Creating and Editing Simulation Models

This section explains how to use the Simulation Object Editor (SOE) to create and edit simulation model sets and the simulation models used by simulation objects. It also covers configuration files that you must edit by hand.



68. The Simulation Object Editor and SMSs

The Simulation Object Editor is a standalone tool that you can use to configure simulation objects and tactical graphics, and also create new simulation model sets.

68.1. Introduction to the Simulation Object Editor 1305
68.2. Starting the Simulation Object Editor 1305
68.3. Simulation Model Sets 1309
68.3.1 Including Simulation Model Sets in Other Simulation Model Sets 1310
68.3.2 Enabling and Disabling Read Only Status for SMSs 1311
68.3.3 SMS Priority for Included and Multiple SMSs 1312
68.3.4 Lua Scripts in Included SMSs 1313
68.3.5 Application-Level Files in Included SMSs 1314
68.3.6 Search Paths for Referenced Files in an SMS 1314
68.3.7 Hiding Included Objects 1314
68.3.8 Promoting a Model to the Derived SMS 1314
68.3.9 Promoting a System to the Derived SMS 1316
68.4. Visual Model Definitions 1316
68.4.1 User-Created Visual Definition Files 1317
68.5. Upgrading Old Simulation Model Sets 1318
68.5.1 The Old SMS Includes a Standard SMS 1318
68.5.2 The Old SMS Was Copied from a Standard SMS 1319
68.5.3 The Old SMS Was Completely New or Copied from base.sms 1320
68.5.4 Copying Visual Data 1320
68.5.5 Running the SMS Upgrade Tool 1321
68.5.6 Migrating an SMS to a New Release by Hand 1321
68.5.7 Migrating Pre-4.7 Weapon and Damage Systems 1322
68.5.8 Converting Pre-4.8 SMSs To Use Articulated Parts 1323
68.6. Opening a Simulation Model Set 1323
68.7. Creating New Simulation Model Sets 1323
68.7.1 Creating a New SMS that Includes an Existing SMS 1324

68.7.2 Creating a New SMS with Minimal Objects 1325
68.7.3 Creating a New SMS by Copying an SMS 1325
68.8. Importing .entity Files 1326
68.9. Applying New and Updated Variables 1326
68.10. Importing and Exporting Object Information 1326
68.10.1 Exporting Object Information 1327
68.10.2 Importing Object Information 1327
68.11. Editing the Tags List 1327
68.11.1 Adding a New Tag 1327
68.11.2 Removing a Tag 1329
68.11.3 Renaming a Tag 1329
68.11.4 Undoing Tag Changes 1329
68.11.5 Using Auto Tags 1330
68.11.6 Assigning Tags Based on Visual Model Characteristics 1330
68.11.7 Exporting Tags 1330
68.11.8 Importing Tags 1331
68.11.9 Managing Tags Using the Object Information File 1331
68.12. Managing the Forces List 1331
68.12.1 Adding a Force 1331
68.12.2 Editing a Force 1333
68.12.3 Removing a Force 1333
68.13. Copying the Object List to the Clipboard 1333

68.1. Introduction to the Simulation Object Editor

Using the Simulation Object Editor (SOE), you can create and edit simulation model sets and the objects in them.

Many VR-Forces users want to edit some or all of these simulation object characteristics:

- Weapon systems
- Basic parameters
- Sensors
- 2D icons and 3D models
- Formations

Users also want to add new simulation objects or tactical graphics types. The Simulation Object Editor allows you to edit most of the aspects of a simulation object or tactical graphic that a typical VR-Forces user will want to change. You can easily add and remove sensors, weapon systems, and other major component systems, change the 2D and 3D models used to represent them, and edit their properties.

In addition to editing simulation objects, you can use the Simulation Object Editor to view information about and configure some types of systems.

The Simulation Object Editor does not allow you to add or remove individual components or edit some of the more detailed entity parameters. For these advanced editing tasks, you must edit specific configuration files, which are described throughout this manual.

The Simulation Object Editor primarily allows you to edit properties that affect simulation. Simulation objects also have visualization models, which are configured in the GUI in the Visual Model Editors and Simulation Object Type Mappings dialog boxes. However, you can also use the Simulation Object Editor to configure the 3D models and 2D icons used to display simulation objects.

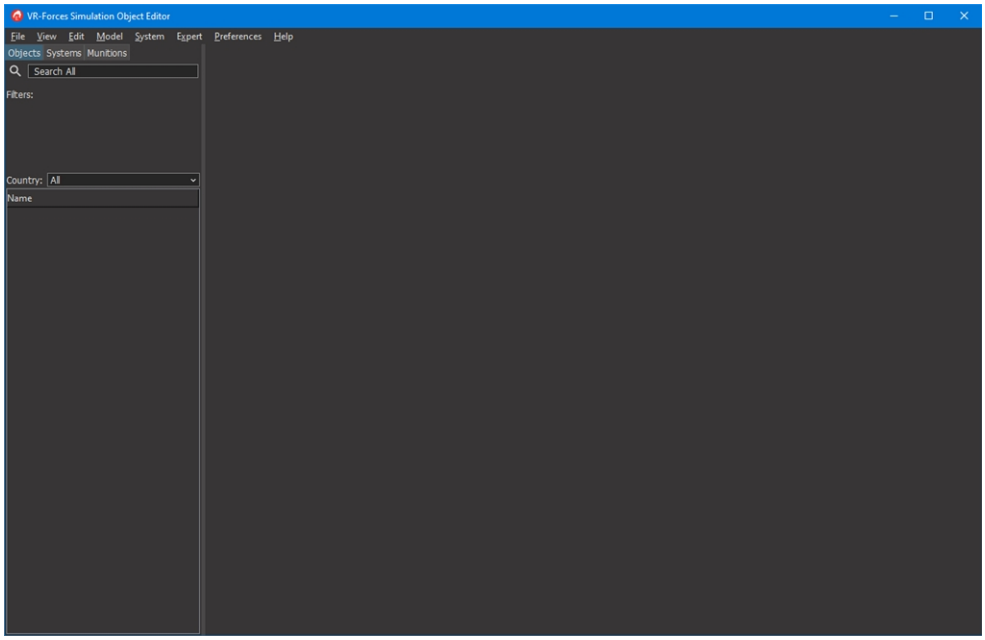


- To edit particle systems, such as smoke and explosions, you use the Particle System Editor. See the *Special Effects (Particle Systems)* chapter in the *Adding Content to MAK ONE Applications Reference Manual*.
- The Expert menu contains items that are beyond the scope of this documentation and are not fully supported. Those features are intended for use by highly knowledgeable end users, including employees of MAK Technologies.

68.2. Starting the Simulation Object Editor

➤ To start the Simulation Object Editor in Windows, choose **MAK Technologies > VR-Forces #.# Simulation Object Editor**. The Simulation Object Editor opens (Figure 538).

Figure 538. Simulation Object Editor



➤ **To start the Simulation Object Editor from the command-line in Windows or Linux:**

1. Change directory to *./bin64*.
2. Run: `simulationObjectEditor` *`options`*

Table 55 describes the command-line arguments for the Simulation Object Editor.

Table 55. Simulation Object Editor command-line arguments

Argument	Description
`--appDataDir` *`directory`*	Specifies the location of application data. If not specified, VR-Forces uses the default: *./appData*.
`--dataDir` *`directory`*	Specifies the location of the application's data directory. If not specified, VR-Forces uses the default, *./data*.
`--doNotCheckPluginVersions`	Do not check versions in the plug-ins to make sure they can load against this version of the application.
`--doNotLoadVrfPlugins`	Prevents loading of plug-ins in the *./plugins* directory.
`(--locale | -G)` *`language`*	Specifies the locale (for localization). Use two-character country code.

Table 55. Simulation Object Editor command-line arguments (continued)

Argument	Description
`--gui` *`filename`*	Specify a GUI configuration file. This file specifies the layout of menus, toolbars, and dialog box pages. A simple filename assumes *./appdata/settings/vrfGui*. A relative filename is from the *./bin*64 directory. A full path is used as is.
`--launcherLocation` *`location`*	Specifies the location of the VR-Forces launcher.
`--layoutSettingsFile` *`filename`*	Specifies the GUI layout settings file to use. *`filename`* specifies the base filename. It is prefixed with *default_* and the extension *.uisx* is added. Default: SOEUILayoutSettings.
`--lightMode`	The SOE loads in light mode rather than the default of dark mode.
`--limitEntityTextures "`*`width height`*`"`	Uses only mipmap levels at or below the resolution for entities. Use two integers for the maximum width and height of the texture mipmap to use, for example, `"1024 1024"` or `"2k 2k"`.To turn off the texture limit function, use `--limitEntityTextures "-1 -1"`.
`--limitEntityTexturesExclusions "`*`texture`*`"`	Excludes textures from texture reduction limits for indirect. Specify the texture(s) that you want to exclude, for example, `--limitEntityTexturesExclusions "SurfaceCVN-75HarrySTruman, SurfaceCruiseShip, FixedWingF-16, FixedWingRQ-1Predator"`.

In most situations, using `--limitEntityTextures` and `--limitTerrainTextures` prevents crashes with high VRAM. If you continue to experience crashes, use `--limitGpuMemoryPercentage` in addition to these command-line options.

`--limitGpuMemoryPercentage` *`integer`*	Set the cutoff percentage as a value between 0 and 100. (Typical values are between 85 and 100.) Beyond the specified percentage, VR-Forces does not make new textures or buffers as a percentage of the available video RAM.
`--loadPlugin` *`filename`*	Load the specified Simulation Object Editor plugin. Can be a specific plug-in or plug-in XML file.

Table 55. Simulation Object Editor command-line arguments (continued)

Argument	Description
(--notifyLevel | -n) *notification_level*	Specifies the notification level for application messages, where *notification_level* is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2. For details, see "5.4.3 Setting the Notification Level" on page 189.
--settingsDirectory *filepath*	Sets the directory to load all UI and settings files from.
--sharedDataDir *directory_path*	Use this to specify the location of the shared data directory if you installed it somewhere other than the default, which is *./SharedData*.
--sharedDataRoot *directory_root*	Specifies the location of the shared data root. This combined with the stem and data version creates the shared data path.
--sharedDataStem *directory_stem*	Specifies the location of the shared data stem. This combined with the data root and data version creates the shared data path.
--sharedSettingsDirectory *directory*	Specifies the directory where shared settings are stored. The default directory is *$(APP_DIR)/settings*. If the --useUserSettingsDirectory flag is set, then the user settings directory is used to store the settings instead.
(--showConsole | -c)	Display the console window.
--smsFile *SMS_name*	Loads the specified SMS file.
--upgrade *SMS_name*	Upgrades the specified SMS in place to the new version. Adding new variables and performing whatever upgrades to the files can be performed without a base SMS to reference
--upgradeSystems *SMS_name*	Upgrades systems in place for the specified SMS to the new version. Adding new variables and performing other upgrades to the files can be performed without a base SMS to reference.
--useUserSettingsDirectory	Indicates that the system should store settings in the user login settings directory, rather than the application settings directory. Use in conjunction with --sharedSettingsDirectory.
--visualTerrain *string*	Terrain used when displaying simulation objects.

Table 55. Simulation Object Editor command-line arguments (continued)

Argument	Description
`(--ignore_rest | --)`	Ignores all command-line options that are listed after this argument. This option is useful for disabling part of a command line in a script for test purposes so that you do not have to retype all of the command options at a later point.
`(--version | -v)`	Displays version information.
`(--help | -h)`	Displays command-line usage information in a console window.

68.3. Simulation Model Sets

A simulation model set (SMS) defines all of the simulation objects, tactical graphics, and other object models that you can create in a VR-Forces simulation. The simulation model set file tells VR-Forces where to find all of the configuration files it needs to create, simulate, and display simulation objects and other objects. Every scenario must reference one or more simulation model sets.

If you use the Simulation Object Editor to edit simulation object models, the only configuration file that you need to be aware of is the simulation model set file that you are using. All of the other files are handled by the Simulation Object Editor.

VR-Forces includes these SMSs:

- *base.sms*. Defines platforms and graphics used by other SMSs.
- *EntityLevel.sms*. Defines entities, units, and other objects used for entity-level scenarios. Includes *base.sms*.
- *AggregateLevelBase.sms*. Defines general units and other objects used by other aggregate-level SMSs. Includes *base.sms*.
- *AggregateLevel.sms*. Legacy SMS preserved for backward compatibility. Includes *base.sms* and *AggregateLevelBase.sms*.
- *AggregateTacticalLevel.sms*. Defines aggregate-level warfare units and other objects suitable for tactical-level scenarios using battalion-size simulation objects or smaller. Includes *base.sms* and *AggregateLevelBase.sms*.
- *MAKTest.sms*. Defines some fictional entities for testing entity-specific features such as lighting. Includes *base.sms* and *EntityLevel.sms*.

For details about entity- and aggregate-level modeling, see "13.7. Entity-Level Modeling and Aggregate-Level Modeling" on page 368.

> ⚠ When you use the Simulation Object Editor, you are editing a simulation model set. We strongly recommend that you keep the default SMSs as reference model sets and not edit them. If you want to edit entity models, create a new SMS that includes one of the provided SMSs. Promote the objects you want to edit to the new SMS and use it as the default SMS for your scenarios. For details about creating a new SMS and changing the default SMS, see "68.7. Creating New Simulation Model Sets" on page 1323.

68.3.1 Including Simulation Model Sets in Other Simulation Model Sets

VR-Forces simulation model sets can include other SMSs. This allows you to create core SMSs that can be included in multiple other SMSs that may have incompatible characteristics or which override some characteristics of the core SMS, but which still need core objects. VR-Forces provides several SMSs, described in "68.3. Simulation Model Sets" on the previous page.

It is important to understand the difference between using an SMS that includes other SMSs and creating a scenario that uses multiple SMSs. Since VR-Forces users usually create new SMSs to configure new types of simulation objects, or add new systems to existing objects, we will consider the differences in that context.

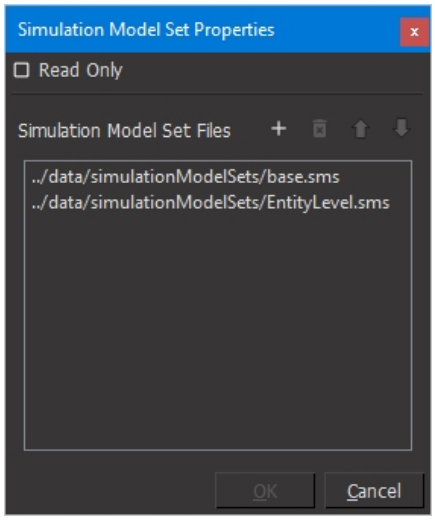
If you want to create a custom standalone SMS for a new or enhanced object, that SMS must include all of the files required to support that object, such as an OPE file, entity file, system files, damage files, and so on. When trying to build such an SMS, it can be easy to overlook required files. This process also requires you to duplicate files that are contained in the supplied SMSs, even if you do not need to change them, because the SMS must contain all the files it needs. Then, if you do not load the SMSs in the correct order, your customization can be overridden by one of the other SMSs you are using in the scenario. In summary, it is a cumbersome process that can create needless duplication of files and be difficult to debug if problems arise.

Using included SMSs minimizes the problems inherent in using multiple standalone SMSs. When an SMS includes another SMS, it has a higher priority than the included SMS. When an SMS is loaded, included SMSs are loaded first, and each subsequent SMS has a higher priority than the previously loaded SMS. Any parameters or files in the higher priority SMSs that are also in the lower priority SMSs override those in the lower priority SMSs. The higher priority SMSs do not need to contain any files that they do not change from the included SMSs. Therefore the total footprint of the SMS is smaller. For more information about SMS priority, see "68.3.3 SMS Priority for Included and Multiple SMSs" on page 1312.

➤ **To include an SMS in another SMS:**

1. Choose **MAK Technologies > VR-Forces 5.2 Simulation Object Editor**. The Simulation Object Editor opens (Figure 538).
2. Choose **File > Open Simulation Model Set**. The Open Model Set dialog box opens.
3. Select the SMS where you want to include another SMS.
4. Click Open. The SMS loads.
5. Choose **File > Properties**. The SMS Properties dialog box opens.

Figure 539. SMS Properties dialog box



6. Click the Add button (+). A line is added to the window with the text <empty file>.
7. Click the Browse button at the end of the line. The Choose File dialog box opens.
8. Select the SMS that you want to include.
9. Click Open. The SMS is added to the list in the window.
10. Click OK. The loaded (derived) SMS is saved and reloaded.

68.3.2 Enabling and Disabling Read Only Status for SMSs

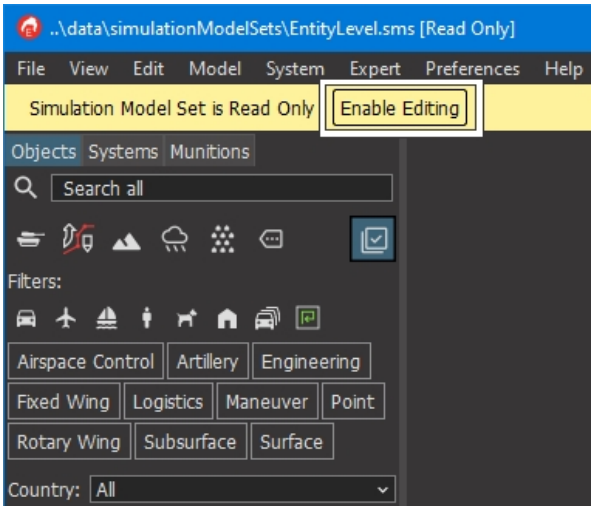
By default, the SMSs provided with VR-Forces are read-only. You can enable and disable editing.

> Read-only status is to encourage you to create new SMSs for your customizations to the SMSs provided by VR-Forces. Leaving the default SMSs unedited simplifies upgrades to future versions of VR-Forces.

➤ **To enable editing of the objects in an SMS, do one of the following:**

- Click the Enable Editing button (Figure 540). You are prompted to create a new SMS that includes the current one, to enable editing for this and future sessions, or to enable editing for this session only. If the current SMS is one of the three basic SMSs provided by VR-Forces, we recommend that you create a new SMS.

Figure 540. Enable Editing



- Choose **File > Properties** and clear the Read Only check box in the SMS Properties dialog box.

➤ **To enable read-only status for an SMS:**

1. Choose **File > Properties**. The SMS Properties dialog box opens.
2. Select the Read Only check box.

68.3.3 SMS Priority for Included and Multiple SMSs

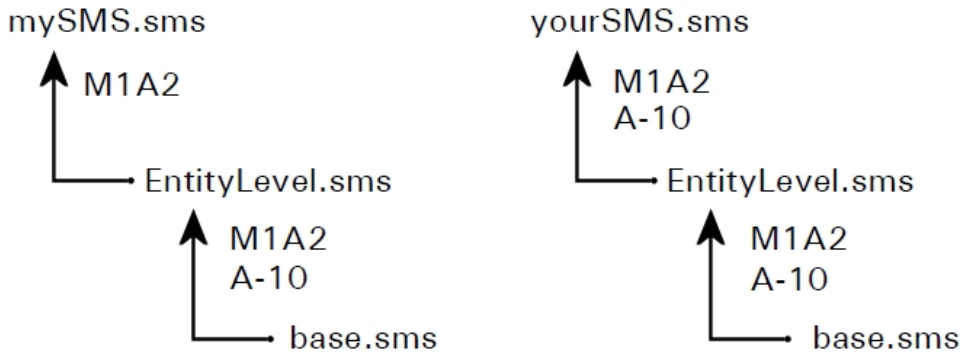
When you load multiple SMSs, either through inclusion or by specifying multiple SMSs for a scenario, conflicts are resolved through the priorities of the SMSs.

For included SMSs, priority is based on the inclusion relationships. The highest level SMS is loaded last and has the highest priority. Simulation objects in higher priority SMSs override those in lower priority SMSs. For example, suppose you create a new SMS called *mySMS.sms* that includes *EntityLevel.sms*. *EntityLevel.sms* already includes *base.sms*. When VR-Forces loads mySMS, it loads *base.sms* first, then *EntityLevel.sms*, and finally *mySMS.sms*. Objects in *mySMS.sms* override objects in *EntityLevel.sms* that have the same object type, which in turn overrides *base.sms*.

When you load multiple SMSs, priority is based on the order in which they are loaded. Assuming a simple case of two SMSs that do not include other SMSs, if the second SMS loaded specifies objects that are also in the first SMS loaded, the specification in the second SMS takes precedence.

If you load multiple SMSs that include other SMSs, priority is based on the order loaded, but duplicate SMSs are not loaded twice. Assume the SMS organization shown in Figure 541.

Figure 541. Multiple SMSs



If *mySMS.sms* is the first SMS loaded, VR-Forces loads *base.sms*, then *EntityLevel.sms*, then *mySMS.sms*. Priority is determined as described previously for an SMS with included SMSs. The M1A2 entity configured in *mySMS.sms* overrides the one in *EntityLevel.sms*.

Then VR-Forces loads *yourSMS.sms*. It starts with *base.sms*. Since *base.sms* is already loaded, VR-Forces ignores it. Then it moves to *EntityLevel.sms*, which is also already loaded and therefore VR-Forces ignores it, as well. Then it loads *yourSMS.sms*. The M1A2 in *yourSMS.sms* overrides the configuration in *mySMS.sms*. The A-10 in *yourSMS.sms* overrides the A-10 in *EntityLevel.sms*.

The overall priority for the SMSs in this example, from lowest to highest, is *base.sms*, *EntityLevel.sms*, *mySMS.sms*, *yourSMS.sms*.

The priority of multiple SMSs is determined by the order in which they are listed in the **Simulation-Model-Set-Files** parameter in the scenario file. In the following example, *EntityLevel.sms* has the highest priority. When you create a scenario and list multiple SMSs, the priority in the SMS list is top to bottom.

```
  (Simulation-Model-Set-Files
    "..\data\simulationModelSets\EntityLevel.sms"
    "..\data\simulationModelSets\developer_toolkit_
examples\addTask.sms")
```

68.3.4 Lua Scripts in Included SMSs

System scripts are stored in simulation model sets. When you load an SMS that includes another SMS, the scripts in the highest priority SMS supersede those in the lower priority SMSs.

All simulation objects use the highest priority scripts, regardless of which SMS they are in. For example, suppose *EntityLevel.sms* has a script called Patrol Ground Area and *mySMS.sms* (the higher priority SMS) also has a script called Patrol Ground Area. If the entity M1A2, which is defined in *EntityLevel.sms* executes the Patrol Ground Area task, it uses the version in *mySMS.sms*.

Lua scripts in included SMSs have additional relationships:

- Any scenario scripted task promoted to a system scripted task is promoted into the highest-priority SMS.

- If you edit a scripted task that is in an included SMS, it is saved into the highest-priority SMS. The version of the script in the included SMS does not change. Therefore if you want a change to a script to affect all uses of that script, edit it in its own SMS.

68.3.5 Application-Level Files in Included SMSs

Application-level SMS files, such as *commModelParams.mtl* or *gui/initialCreateMenuConfiguration.xml*, are included the same way entity-specific files are and you do not need to have copies of them in an derived SMS as long as one of the included SMSs has them. However, every SMS must have *vrfSim.opd*.

68.3.6 Search Paths for Referenced Files in an SMS

The simulation objects in a scenario reference many files to determine their behavior, such as system definition files, scripts, platform files, and so on. If all the simulation objects in a scenario are configured in one SMS, then all of the referenced files are within the file hierarchy of that SMS. However, if you use an SMS that includes other SMSs, then simulation objects may have to search within multiple SMSs to reference the correct files.

An SMS can load files from any SMS it includes, and any currently loaded SMS it is included in.

File referencing, as with other aspects of included SMSs, is based on the priority of the SMS in which an object is configured. Given the SMSs in Figure 541 on the previous page, the search paths for each SMS from lowest to highest priority would be:

- *base.sms*. Can search in *base.sms*, *EntityLevel.sms*, *mySMS.sms*, and *yourSMS.sms*.
- *EntityLevel.sms*. Can search in *base.sms*, *EntityLevel.sms*, *mySMS.sms*, and *yourSMS.sms*.
- *mySMS.sms*. Can search in *base.sms*, *EntityLevel.sms*, and *mySMS.sms*.
- *yourSMS.sms*. Can search in *base.sms*, *EntityLevel.sms*, and *yourSMS.sms*.

68.3.7 Hiding Included Objects

By default, the Simulation Object Editor lists the simulation objects that are part of included SMSs. However, you can hide included objects if you just want to see the objects in the top-level SMS.

➤ To hide or display included objects in the objects list, choose **Preferences > Show Included Simulation Objects**.

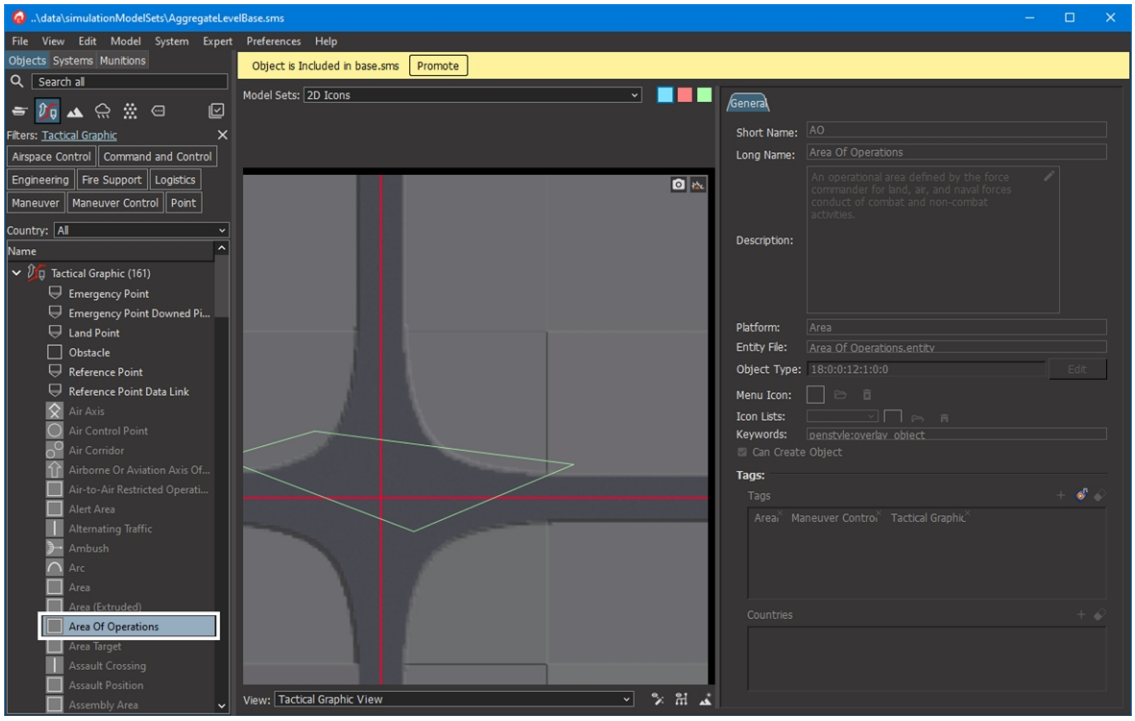
68.3.8 Promoting a Model to the Derived SMS

The preferred approach when you need to customize something from an included SMS is to promote it for edits in the derived SMS.

When you load an SMS that has included other SMSs, the objects from the included SMSs are listed but you cannot edit them. Without promiting them, you can only edit them by editing the SMS to which they belong. Figure 542 shows *AggregateLevelBase.sms* filtered

for tactical graphics. The objects at the end of the list are not editable because they are included from *base.sms*. Note in the figure that most of the parameters for the selected object are read-only.

Figure 542. Included simulation object

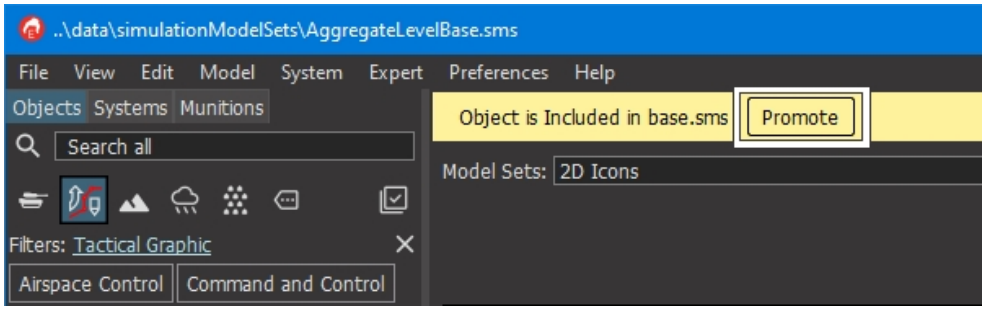


Since one of the purposes of being able to include SMSs in the loaded (derived) SMS is to customize simulation objects in the included SMSs, you can promote objects from the included SMS to the loaded SMS. The object then becomes editable and is saved in the loaded SMS. When the derived SMS is loaded, the object in the loaded SMS overrides the object in the included SMS.

➤ **To promote an object to the derived SMS:**

1. Load the derived SMS.
2. Choose **Preferences > Show Included Simulation Objects** if it is not already enabled.
3. On the Objects tab or Munitions tab, select the object you want to promote. (Although munitions are on a separate tab, they are also simulation objects.)
4. Choose **Model > Promote to Main Simulation Model Set** or click the Promote button (Figure 543) (above the parameter configuration tabs). The object is added to the derived SMS and is now fully editable. When you save the SMS, a new *.entity* file is added to it.

Figure 543. Promote button



68.3.9 Promoting a System to the Derived SMS

Just as with simulation objects ("68.3.8 Promoting a Model to the Derived SMS" on page 1314), if you want to edit a system in an included SMS, you must promote it to the loaded (derived) SMS.

➤ **To promote an object to the derived SMS:**

1. Load the derived SMS.
2. Choose **Preferences > Show Included Simulation Objects** if it is not already enabled.
3. Select the system you want to promote.
4. Choose **System > Promote to Main Simulation Model Set** or click the Promote button (Figure 543 above). The system is added to the derived SMS and is now fully editable.

68.4. Visual Model Definitions

VR-Forces stores visual model definitions in different places depending on which SMS is loaded:

- When the Simulation Object Editor loads one of the SMSs that ship with VR-Forces (for example, *base.sms*, *EntityLevel.sms*, *AggregateLevel.sms*, or *AggregateLevelBase.sms*), it reads the appropriate definition files for the SMS from the directories under *./SharedData/19/latest/ModelData/Configuration/Definitions*. When you save changes to one of these SMSs, the changes are written back to the original definition files.
- If you create a new SMS, model and element definition files are saved with the SMS in the *./data/simulationModelSets/<sms>/gui/visuals* directory for the new SMS. When you load the SMS, it reads model and element definitions from the SMS-specific files.

For information about model and element definition files, see the *Model Definitions* and *Element Definitions* chapters in the *Adding Content to MAK ONE Applications Reference Manual*.

68.4.1 User-Created Visual Definition Files

Although we recommend that you use the Simulation Object Editor for all changes to visual models, it is possible to add new model definitions and element definitions using the Visual Model Editors dialog box instead of the Simulation Object Editor. Additionally, some attributes of element definitions cannot be set in the Simulation Object Editor, so if you want to change them, you must use the Visual Model Editors.

You can export the new and changed definitions to files other than the default set of definition files. In fact, in many cases, we recommend that you export them to their own files. However, creating new model or element definitions in the Visual Model Editors and saving them to new files has implications for how they are loaded and saved by the Simulation Object Editor.

When the Simulation Object Editor loads one of the five default SMSs, it loads all files in the appropriate directories, including any additional files that you have created. However, when VR-Forces saves changes to the SMS, it writes the changes only to the default definition files for the SMS. It does not write back to the user-created definition files. Since the user-created files are still in the directory, the next time you load the SMS, there may be conflicts between the files and the results may be unexpected.

Given this behavior, the recommended workflow for reconciling new definitions created in the Visual Model Editors dialog box is:

1. In the Visual Model Editors dialog box, save new model or element definitions to new files in the default definition directories.
2. Open the affected SMS in the Simulation Object Editor.
3. Confirm that your new model or element definitions have been loaded by the SMS.
4. Save the SMS. This saves the new definitions into the default set of SMS definition files.
5. Delete the individual definition files that you created in step 1 so that there are no conflicts with the SMS.

If you create a new SMS, when the Simulation Object Editor loads the SMS, it only loads the definition files that it previously created. If you create additional leaf or model definition files with the Visual Model Editors dialog box, they are not loaded. You must import them into your SMS. Therefore, the recommended workflow is:

1. Open a scenario that uses your SMS.
2. In the Visual Model Editors dialog box, load the *.hier* file for your SMS. It is either *./SharedData/19/latest/ModelData/Configuration/Definitions /Aggregate/ElementDefinitions/coreAggregateElementDefinitions.hier* or *./SharedData/19/latest/ModelData/Configuration/Definitions /Entity/coreEntityElementDefinitions.hier*.
3. Import the model definition or element definition file to which you want to add new definitions. Be sure to import it from the files in the */data/simulationModelSets/<sms>/gui/visuals* directory for your SMS. Be sure that you only edit one element definition file at a time.

4. Add new definitions as desired.
5. Export the new definitions to the definition file that you imported.
6. Open the SMS in the Simulation Object Editor and confirm that the new definitions are available.

68.5. Upgrading Old Simulation Model Sets

The Simulation Object Editor has an upgrade tool that can upgrade SMSs created in VR-Forces 4.4 or later to the current release. The upgrade process varies depending on how you created the SMS that you are upgrading. The possibilities it takes into account are:

- The old SMS was created by including a standard SMS (*EntityLevel.sms*, *AggregateLevelBase.sms*, or *AggregateLevel.sms*).
- The old SMS was created by copying one of the standard SMSs.
- The old SMS was a new SMS that did not copy or include a standard SMS. It could have included or be a copy of *base.sms*.

The following sections describe the upgrade process for each of these possibilities. If you need to upgrade an SMS from VR-Forces 4.3.x or earlier, or want to upgrade your SMS manually, see "68.5.6 Migrating an SMS to a New Release by Hand" on page 1321.

68.5.1 The Old SMS Includes a Standard SMS

If you created the old SMS by including a standard SMS, the assumption is that you made no changes to the included SMS. You edited your SMS by promoting simulation objects from the standard SMS and editing them, and by adding new simulation objects in your SMS. You may also have made other changes to the SMS, such as adding scripts and adding or editing system definition files.

Since you did not touch the standard SMS, the upgrade tool assumes that you will simply include the standard SMS that is provided in the new version of VR-Forces and that you want to keep all the changes you made when creating the old SMS. The upgrade wizard does the following:

- Copies all files from the old SMS (except visual data as described in the next paragraph).
- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, see "68.5.4 Copying Visual Data" on page 1320.)
- Compares the files that were promoted into the old SMS to files of the same name in the new included SMS. If the files are different, these files are listed as needing investigation.

68.5.2 The Old SMS Was Copied from a Standard SMS

Upgrading an SMS that was created by copying a standard SMS (*EntityLevel.sms*, *AggregateLevelBase.sms*, or *AggregateLevel.sms*) is more complex than upgrading an SMS that was created by including an SMS. The upgrade tool tries to preserve all changes made to the files in the old SMS while incorporating the improvements made to the new version of the standard SMS. However, it errs on the side of keeping changes made to files in the old SMS. This means that if you want your SMS to take advantage of the improvements to the new standard SMS, you may have to compare files by hand. If changes have been made to platform files, you can quickly capture those changes by selecting Apply Platform Changes in the Simulation Object Editor.

The upgrade tool does the following:

- It generates checksums for the files in the old SMS. It compares these checksums to checksums for the files in the standard SMS in the old version of VR-Forces and to the files in the standard SMS in the version of VR-Forces you are upgrading to. It moves files to the new SMS based on these comparisons. Table 56 describes the possible results of these comparisons.

Table 56. Decision table for comparing SMS files

	Yes	Notes
Is the file the same in old standard SMS, old SMS, and new standard SMS?	Copy file to new SMS.	
Is the file the same in old standard SMS and old SMS, but changed in new standard SMS?	Copy file from new standard SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file only in old SMS?	Copy file from old SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file in old standard SMS and old SMS, but removed from new standard SMS?	If the file was changed in old SMS, copy it to new SMS. Otherwise, remove it.	Optionally, run Apply Platform Updates to capture changes in OPE files.
Is the file only in new standard SMS?	Add this file to the new SMS.	

Table 56. Decision table for comparing SMS files (continued)

	Yes	Notes
Did the file change from the old standard SMS to the old SMS and from the old standard SMS to the new standard SMS?	Copy file from old SMS to new SMS.	Consider merging in changes from the file in the new standard SMS.
Is the file the same in old standard SMS and new standard SMS, but different in old SMS?	Copy file from old SMS to new SMS.	Optionally, run Apply Platform Updates to capture changes in OPE files.

- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, see "68.5.4 Copying Visual Data" below.)
- The upgrade tool provides a report of files that have conflicts, files that were added, and files that were removed. You can use this report to determine what files you want to examine for manual upgrades.

68.5.3 The Old SMS Was Completely New or Copied from *base.sms*

In rare cases, VR-Forces users may create an SMS by copying or including only *base.sms*, by selectively copying files from one of the standard SMSs, or by creating all the files themselves. In this case, there is no reference for the upgrade process. The upgrade tool does the following:

- Copies all files from the old SMS (except visual data as described in the next paragraph).
- If the old SMS is from VR-Forces 4.5 or later and you specify in the upgrade wizard that you want to copy visual data, it is copied to the new SMS. If you specify that you do not want to copy visual data, no visual data files are copied. (For more information about copying visual data, see "68.5.4 Copying Visual Data" below.)
- Reports files that might need to be reviewed, such as system definitions and OPE files.

68.5.4 Copying Visual Data

In VR-Forces 4.4.x, all visual model data was stored in files in *./appData/definitions* and in *./appData/settings/vrfGui*. Visual data was not specific to an SMS. In VR-Forces 4.5 and later, visual data files were restructured and an SMS can have visual data that is specific to the SMS.

For SMSs created in VR-Forces 4.5 or later, the upgrade tool gives you the option of copying visual data from the old version of VR-Forces. If you choose to copy visual data and the old SMS has SMS-specific visual data, the files from the SMS are copied to the new SMS. If the old SMS did not have SMS-specific data, the files from *./SharedData/19/latest/ModelData/Configuration/Definitions* are copied to the new SMS.

If you choose not to copy visual data, the new SMS uses the visual data provided in the new version of VR-Forces.

If you copy visual data to the new SMS, it will take longer to load the SMS when you open a scenario.

68.5.5 Running the SMS Upgrade Tool

➤ **To upgrade an SMS to a new version of VR-Forces:**

1. Open the Simulation Object Editor.
2. Choose **File > Upgrade Old Simulation Model Set**. The Upgrade Simulation Model Set dialog box opens. (The upgrade process varies, depending on the type of SMS you are upgrading.)
3. Select the SMS that you are upgrading (the old SMS).
4. Specify a name for the upgraded SMS. The default is the name of the old SMS.
5. Click Next.
   - If the old SMS was created by copying a standard VR-Forces SMS or was a completely original SMS, the Original Simulation Model Set page displays.
   - If the old SMS was created by including a standard VR-Forces SMS, the Simulation Model Set Version page displays. Skip to step 8.
   - If the old SMS was created by copying a standard SMS, select the standard SMS it was copied from.
   - If the source is completely original or was copied from *base.sms*, select Neither. (SMSs created by including standard VR-Forces SMSs do not see this page.)
6. Click Next.
7. The Simulation Model Set Version page displays.
8. Select the version of VR-Forces used to create the old SMS.
9. Click Next.
   - If the old SMS was created with VR-Forces 4.4 or 4.4.2, the upgrade process runs and a report is generated.
   - If the old SMS was created with VR-Forces 4.5 or later, the Visual Data page displays.
10. Specify whether to copy visual data.
11. Click Next. The upgrade process runs and generates a report. You can save the report ().
12. When you are done, click Finish.

68.5.6 Migrating an SMS to a New Release by Hand

If you need to migrate an SMS that was created in VR-Forces 4.3.x or earlier or do not want to use the upgrade tool, you can migrate your old SMS by hand. You can do this by copying your customized files to a new SMS or by recreating your changes in a new SMS.

When you edit an SMS in the Simulation Object Editor, it:

- Creates or edits *.entity* files for the simulation objects that you edit.
- Modifies visual definition files and entity type mappings.

The Simulation Object Editor does not create or change any other files. However, there are many other files in an SMS that could be customized, such as system definitions, the various damage and ammo select files, and system scripts.

If you upgrade to a new version of VR-Forces, you can quickly add any of the new simulation objects you created by copying their *.entity* files into an SMS in the new release. This will make the simulation objects available to the simulation engine, but will not include their visual definitions. For each entity that you add this way, you will have to edit the 2D and 3D models in the Simulation Object Editor.

If you made any changes to other files, you may have to copy those changed files to the new SMS.

If you have written scripts, use the script export process to export them from the previous release of VR-Forces and then import them into the new release.

Since SMSs often have significant changes from one release to the next, the older your legacy SMS is, the harder it will be to upgrade it simply by copying changed files. We recommend that you review the release notes for interim releases to understand the changes that you might need to account for when migrating old files. You can view the release notes for each release on the Product Specifications & Dependencies page for VR-Forces at https://www.mak.com/mak-one-specs/vr-forces.

There are several major SMS divergence points to be aware of:

- In VR-Forces 4.2, *default.sms* was reorganized as *base.sms* and *EntityLevel.sms*.
- In VR-Forces 4.5, visual model files were restructured.
- In VR-Forces 4.7, weapon system and damage system files were changed significantly so that they could be edited in the Simulation Object Editor.

68.5.7 Migrating Pre-4.7 Weapon and Damage Systems

Due to changes to the system definition files to support editing weapon and damage systems in the SOE, the SMS upgrade tool does not migrate weapon and damage systems from the VR-Forces 4.6.1 and earlier format to VR-Forces 4.7 or later. If you have modified a system using the legacy file formats and want to replicate it in the current release, you have two options:

- Copy the old system files to the new version of VR-Forces. VR-Forces continues to support the VR-Forces 4.6.1 and earlier format. If you copy the old files to the current SMS, you will not be able to edit them in the SOE.
- Use the SOE to edit an existing system or create a new system to match the system that you customized in the older version of VR-Forces.



There are more details about the changes to weapon and damage systems in the *VR-Forces Migration Guide*.

68.5.8 Converting Pre-4.8 SMSs To Use Articulated Parts

In VR-Forces 4.8, entities' articulated parts were enhanced. You can add and adjust them in the Simulation Object Editor, and, using the Set menu, you can set the appearance of articulated parts that are not managed by another controller. To take advantage of this feature, you must update the *.entity*, *.sysdef*, and *.ope* files as described in the *VR-Forces Migration Guide*.

To learn more about editing articulated parts in the Simulation Object Editor, see "70.3. Editing a Model's Articulated Parts" on page 1387.

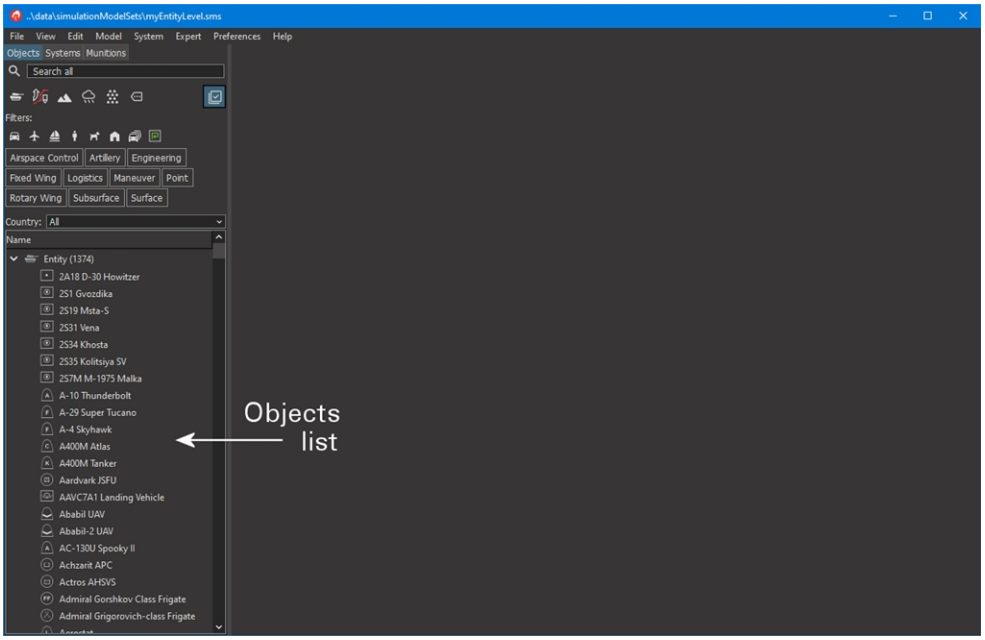
68.6. Opening a Simulation Model Set

To edit simulation object models in the Simulation Object Editor, you must open a simulation model set.

➤ **To open a simulation model set:**

1. Choose **File > Open Simulation Model Set**. The Open Model Set dialog box opens.
2. Select the SMS you want to open.
3. Click Open. A progress bar indicates what files are loading. Once the SMS loads completely, a list of simulation objects displays in the Simulation Object Editor.

Figure 544. Simulation Object Editor with simulation model set loaded



68.7. Creating New Simulation Model Sets

There are a couple ways to create a new SMS. You can create an SMS that contains no simulation objects but includes an existing SMS, or you can create a new SMS that contains only the most basic objects that VR-Forces requires. You can also create an SMS

by copying an existing SMS. In the first two cases, you would add new objects or promote objects from the included SMSs into the derived (loaded) SMS. In the third case, you would edit the simulation objects already in the SMS and add new ones as desired.

To simplify upgrading to future versions of VR-Forces, the recommended choice is to create a new simulation model set by including an existing SMS.

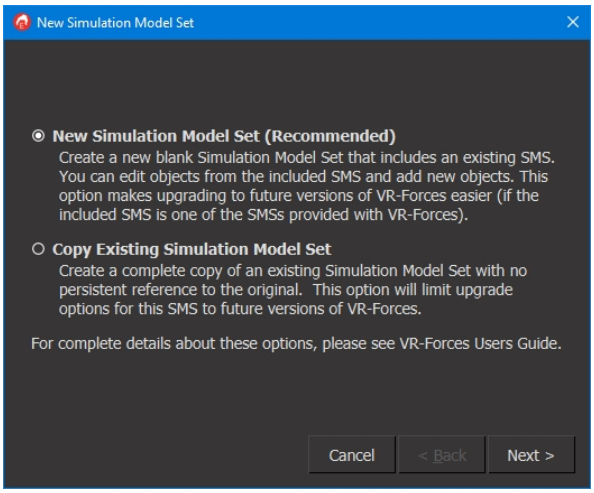
68.7.1 Creating a New SMS that Includes an Existing SMS

The recommended method is to use an existing set that includes models that work in VR-Forces, and then customize it by adjusting the existing models or adding new ones.

➤ **To create a new simulation model set based on an existing one:**

1. Choose **File > New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 545).

Figure 545. New Simulation Model Set dialog box



2. Select New Simulation Model Set.
3. Click Next. The Simulation Model Set Type dialog box opens.
4. Select Extend Existing Simulation Model Set.
5. Click Next. The Extend Existing Simulation Model Set dialog box opens.
6. Select the SMS to use as the basis for your new SMS.
7. Click Next. The New Model Set Name dialog box opens.
8. In the File name box, enter a name for the new model set.
9. Click Finish. The Simulation Object Editor loads systems information from the selected model set.
10. Edit the SMS as desired.
11. Choose **File > Save Simulation Model Set**.

68.7.2 Creating a New SMS with Minimal Objects

Every SMS must include a basic set of files, such as platform files, GUI configuration files, and system definitions. You ensure that your new SMS has those files by including an existing SMS in it. The minimum set of files is provided by *base.sms*.

➤ **To create a new simulation model set with minimal objects:**

1. Choose **File > New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 545 on the previous page).
2. Select New Simulation Model Set.
3. Click Next. The Simulation Model Set Type dialog box opens.
4. Select Minimal Simulation Model Set.
5. Click Next. The New Model Set Name dialog box opens.
6. In the File name box, enter a name for the new model set.
7. Click Finish. The Simulation Object Editor loads basic object information.
8. Edit the SMS as desired.
9. Choose **File > Save Simulation Model Set**.

68.7.3 Creating a New SMS by Copying an SMS

You can create a new SMS by copying an existing SMS. When you do this the new SMS includes any SMSs that were included by the original SMS. All of the simulation objects in the new SMS are editable.

➤ **To create a new simulation model set by copying an SMS:**

1. Choose **File > New Simulation Model Set**. The New Simulation Model Set dialog box opens (Figure 545 on the previous page).
2. Select Copy Existing Simulation Model Set.
3. Click Next. The Select Model Set to Copy dialog box opens.
4. Select a model set and click Next.
5. Click Next. The New Model Set Name dialog box opens.
6. In the File name box, enter a name for the new model set.
7. Click Finish. The Simulation Object Editor loads information from the source model set.
8. Make any changes you want to the model set. (For details about how to add and remove included SMSs, see the procedure in "68.3.1 Including Simulation Model Sets in Other Simulation Model Sets" on page 1310, starting with step 6 (page 1310).)
9. Choose **File > Save Simulation Model Set**.

68.8. Importing .entity Files

If you have *.entity* files that you created with a previous version of VR-Forces or by multiple persons on different computers, you may want to import them into an SMS. The easiest way to do this is to copy the files into the *vrfSim* directory in the SMS (*./data/simulationModelSet/<model_set>/vrfSim*). However, you can also import the files from the Simulation Object Editor.

➤ **To import *.entity* files into an SMS:**

1. Choose **File > Add Entity File**. The Add Entity File dialog box opens.
2. Select the file that you want to import.
3. Click Open. The file is copied to the SMS.

68.9. Applying New and Updated Variables

All simulation object models reference one of the basic platforms, such as ground, fixed-wing, and point object. Most simulation objects reference one or more system definition files. If you update the variable bindings in a platform file or system definition file, you must apply these changes to the simulation objects of that platform type. Although you could do this by hand, as described in "75.5. Adding Variable Bindings to Platform and System Files" on page 1486, that would be a tedious and error-prone activity. The Simulation Object Editor can apply all updates to all files automatically.

Similarly, if new variables are added to the API in descriptor files, these variables must be added to the system definition files and OPE files.

➤ **To apply new and updated variable bindings:**

1. Choose **File > Regenerate SMS Files**. The Regenerate SMS Files dialog box opens.
2. Select the check box for each update you want:
   - Entity File Updates. All new variable bindings in the OPE files and systems are added as elements in the appropriate *.entity* files.
   - OPE File Updates. Applies changes to platform file descriptors to the OPE files.
   - System File Updates. Updates system files and system template files with any new variables that have been added to the system file descriptors.
3. Click OK.

68.10. Importing and Exporting Object Information

You can export a list of all simulation objects in an SMS to a CSV file. You can open the CSV file in a spreadsheet application. You can edit some of the object information and then import the updated object information back into the SMS. This format lets you quickly edit information without selecting objects one-by-one in the Simulation Object Editor.

For each object in the SMS, the object information file lists the object type enumeration, name, short name and filtering tags. You can edit the name, short name and tags.

> ⚠ Although you can also edit the object type, doing so will break the relationship between the limited information in the spreadsheet and the complete configuration stored in the SMS. It is strongly recommended that you not edit the object type.

68.10.1 Exporting Object Information

➤ **To export object information:**

1. Choose **Model > Export Object Information**. The Export Object Information dialog box opens.
2. Enter a name for the file.
3. Click Save.

68.10.2 Importing Object Information

➤ **To import object information:**

1. Choose **Model Import Object Information**. The Import Object Information dialog box opens.
2. Select the file you want to import.
3. Click Open.

68.11. Editing the Tags List

Tags allow you to organize and filter simulation objects in VR-Forces. You can also search by tag. If you want to organize simulation objects into tags that are not included in the default tags list, you can add new tags. You can also remove and rename tags. Tags for simulation objects are stored in *./data/simulationModelSets/<sms>/gui/tags.csv*. Tags for systems are stored in *./data/simulationModelSets/<sms>/gui/system-tags.csv*.

> ⓘ The procedures in this section also apply to editing the Country list. Select Country on the Edit menu instead of Tags.

68.11.1 Adding a New Tag

When you add a new tag, you can give it a rank. Ranks determine how tags display in the tag window:

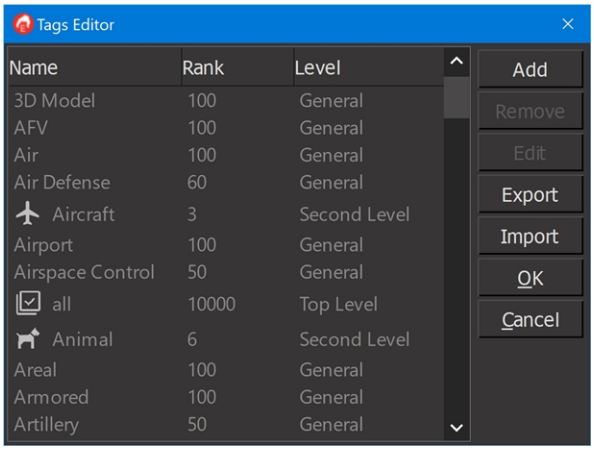
- If a tag is a top-level tag, the rank determines the order in which it is listed relative to other top-level tags. The default top-level tags included with VR-Forces are Clutter (), Entity (), Simulation Effect Object (), Tactical Graphic (), Terrain (), Weather (), Uncategorized, and All.
- If a tag is not a top-level tag, the rank determines the preference for including the tag in the tag list. The width of the Create Object Palette determines the number of tags that display, and then there are further rules determining which tags display. If a top level tag has more possible sub-tags than will fit in the available space, VR-

Forces sorts the tags by rank order starting with 1. Within each group of tags with the same rank, it sorts them based on the number of simulation objects using that tag. Once it determines the tags to display, it lists them in alphabetical order. If you adjust the width of the Create Object Palette, the list updates dynamically.

➤ **To add a new tag:**

1. Choose **Model > Tags**. The Tags Editor opens (Figure 546).

Figure 546. Tags Editor



2. Click Add. The Add Tag Information dialog box opens.
3. Enter a name for the new tag.
4. Optionally, specify a rank.
5. From the Top Level Tag list, select the level where you want the tag to display:
   - Top Level. Displays with other top-level tags such as Entity, Tactical Graphic, and so on, and typically uses an icon.
   - Second Level. Displays with other second-level tags for the appropriate top-level tag and as sorted. Second-level tags are those found immediately below the top-level tags and typically use icons, for example, the Ground Vehicles, Aircraft, and other tags for entities.
   - General. Displays only as text below the second-level tags.
6. To make the tag available for people to select in the GUI, select the Visible check box. If this check box is not selected, the tag is not shown in the GUI but you can still use the tag for organization and also look for it using the Search box.
7. Optionally, select the Organize On List check box to indicate that the tag should be used as a group in the VR-Forces Create Object Palette and the SOE Object list.
8. To use an icon instead of the tag name, click the Add Icon button (+). The Select Tag Icon dialog box opens.
9. Select an icon file.
10. Click Open.
11. Click OK.

- If you add a tag and do not assign a simulation object to it, it does not display in the Create Object Palette even if there is space for it.
- In the VR-Forces Create Object Palette or the SOE Object list, if two top-level tags have the same rank and you select a tag, one of the top-level tags with the duplicated rank is hidden.

68.11.2 Removing a Tag

If you remove a tag, it is removed from any entity that has it assigned. You must save the SMS to make the change permanent.

➤ **To remove a tag:**

1. Choose **Model > Tags**. The Tags Editor opens.
2. Select the tag that you want to remove.
3. Click Remove.
4. Click OK.

68.11.3 Renaming a Tag

When you rename a tag, all simulation objects that are assigned to the old name are reassigned to the new name. The old name remains in the Tags list until you save the simulation model set, but it is not listed in the Tags Editor. If you assign a simulation object to the old tag name before saving the SMS, the old name is retained and added to the list in the Tags Editor.

➤ **To rename a tag:**

1. Choose **Model > Tags**. The Tags Editor opens.
2. Select the tag that you want to rename.
3. Click Edit. The Edit Tag Information dialog box opens.
4. Enter the new name.
5. Click OK. The tags list updates.
6. Click OK.

68.11.4 Undoing Tag Changes

You can undo changes made to the tag list and entity tag assignments.

⚠ When you undo tag changes, all changes to the tags list and all changes to entity tag assignments made since the last save of the simulation model set are reversed. Be sure that this is what you want before completing this procedure.

➤ **To undo tag changes:**

1. Choose **Model > Revert Tags**. A confirmation prompt is displayed.
2. Click Yes.

68.11.5 Using Auto Tags

You can automatically assign a set of tags to new models, to specific models, and to all objects. The tags are assigned by object type based on the information in *./appData/settings/simulationObjectEditor/autoTagInformation.csv*. (The Simulation Object Editor reads the file when it starts up, so if you edit the file while the SOE is open, you must restart the SOE to pick up the changes.)

When you create a completely new model (not a new model from an existing model), if the object type has an entry in *simulationObjectEditor.csv*, the associated tags are automatically added to the model. (When you create a new model from an existing model, the new model has the same tag list as the model it was created from.)

➤ To assign auto tags to the entire object list in an SMS, choose **Model > Apply Auto Tags**.

➤ **To apply auto tags to a specific model:**

1. Select the model whose tags you want to update.
2. Click the Apply Auto Tags button () above the Tags window.

68.11.6 Assigning Tags Based on Visual Model Characteristics

You can automatically assign tags to simulation objects that have 3D models based on their visual characteristics, such as articulated parts and display states. This process examines every visual model and can take a long time. You can use the filter and check boxes to limit the list to only the models that you want to process.

➤ **To assign tags based on visual model characteristics:**

1. Choose **Model > Assign Tags Based on Visuals**. A message informs you of what the process entails.
2. Click Yes. The Select Visual Models dialog box opens. It lists all of the simulation objects in the SMS. The list does not include models that are inherited from an included SMS.
3. Optionally, use the Filter to include only the objects you want to process.
4. Optionally, use the check boxes, Select All, or Clear All to narrow down the list.
5. When the desired objects are selected in the list, click OK.

   The SOE displays a message as it processes the simulation objects and assigns tags. The message closes when the process completes.

68.11.7 Exporting Tags

You can export the tags in an SMS. They are saved to a comma-separated values (CSV) file. You can then import the tag list into another SMS or another VR-Forces installation.

The tag list lists of tags, priorities, and icons associated with the tags, if any. It does not include any information about simulation objects.

➤ **To export the tag list:**

1. Choose **Model > Tags**. The Tags Editor opens.
2. Click Export. The Export Tag Information dialog box opens.
3. Specify a name for the tag file.
4. Click Save.

68.11.8 Importing Tags

➤ **To import a tag list:**

1. Choose **Model > Tags**. The Tags Editor opens.
2. Click Import. The Import Tag Information dialog box opens.
3. Select the tag file you want to import.
4. Click Open.

68.11.9 Managing Tags Using the Object Information File

If you want to edit the tags for many simulation objects, including adding new tags, editing individual simulation objects one-by-one can be tedious. You can speed up the process by editing them in a spreadsheet. To do this:

1. Export the simulation object list to a CSV file, as described in "68.10.1 Exporting Object Information" on page 1327.
2. Make the changes you want to tag assignments.
3. Import the edited object information file into the SMS.

68.12. Managing the Forces List

When you create an object in the GUI, you must choose a force. You manage the list of forces available and their coloring in the Simulation Object Editor.

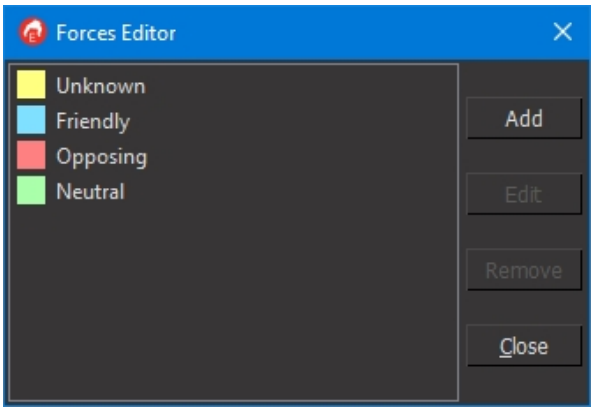
68.12.1 Adding a Force

Adding new forces may be useful if you are simulating participants from multiple countries or if you want to easily model and change force relationships for non-governmental forces such as militias or insurgent groups.

➤ **To add a force:**

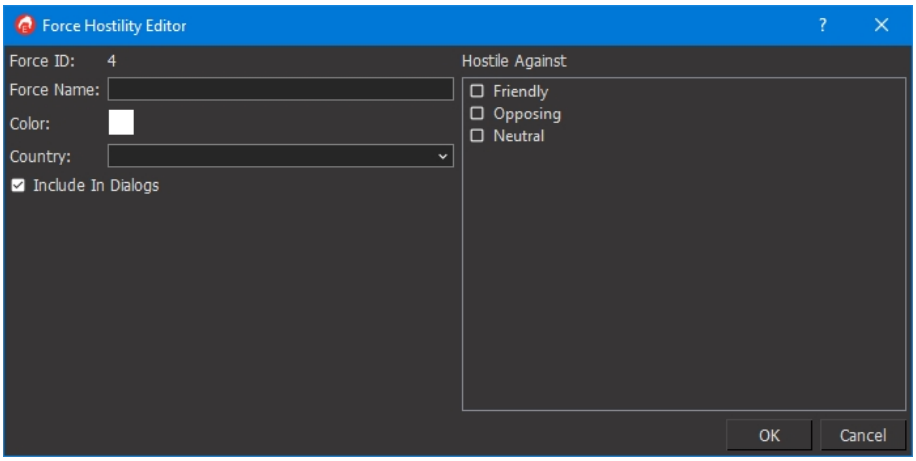
1. Choose **Edit > Forces**. The Forces Editor opens (Figure 547).

Figure 547. Forces Editor



2. Click Add. The Force Hostility Editor opens (Figure 548). The new force is assigned a force ID. You cannot edit the force ID.

Figure 548. Force Hostility Editor



3. In the Force Name box, type a name for the force.
4. Click the Color button. A color picker opens.
5. Select a color for the force.
6. Optionally, in the Country list, specify a country to associate with the force.
7. Optionally, in the Hostile Against pane, select the forces that the new force will be hostile towards. (You can change hostility at run time in the Hostility Matrix. For details, see "9.3. Managing Force Hostility Relationships" on page 300.)
8. If you want the force to be listed in dialog boxes that list forces, select the Include in Dialogs check box.
9. Click OK.
10. Click Close.

68.12.2 Editing a Force

You can edit a force's name, color, hostility, and use. Each force has a force number. When you edit a force, you are changing the attributes of that numbered force.

➤ **To edit a force:**

1. Choose **Edit > Forces**. The Forces Editor opens (Figure 547 on the previous page).
2. Select the force that you want to edit.
3. Click Edit. The Force Hostility Editor opens (Figure 548 on the previous page).
4. To edit the name, change the text in the Force Name box.
5. To edit the color associated with the force, click the Color button and choose a color from the color picker.
6. To edit hostility relationships, in the Hostile Against window, select the forces that the force should be hostile towards.
7. To edit whether the force is listed in dialog boxes that list forces, select or clear the Include in Dialogs check box.
8. Click OK.
9. Click Close.

68.12.3 Removing a Force

If you create a scenario that has simulation objects with a particular force and you then remove the force, the next time you open the scenario, VR-Forces still maintains that force assignment for the simulation objects. The force will be listed in the Hostility Matrix. However, entity icons will not be colored and you will not be able to create any simulation objects with the removed force.

➤ **To remove a force:**

1. Choose **Edit > Forces**. The Forces Editor opens (Figure 547 on the previous page).
2. Select the force that you want to remove.
3. Click Remove. The force is removed from the list.
4. Click Close.

68.13. Copying the Object List to the Clipboard

You can copy the object list to the clipboard and paste it as text into word processing applications.

➤ To copy the object list to the clipboard, choose **Edit > Copy Names in List to Clipboard**.



69. Editing Common Parameters

This chapter explains how to use the Simulation Object Editor to edit simulation object parameters that are common to most simulation object types for both entity-level scenarios and aggregate-level scenarios. For details about editing parameters that are specific to individual entities, see "Chapter 70. Editing Entities for Entity-Level Scenarios" on page 1375. For details about editing units, see "Chapter 71. Editing Units for Entity-Level Scenarios" on page 1405 and "Chapter 72. Editing Aggregate-Level Objects" on page 1417.

69.1. Editing Generic Object Parameters 1337
- 69.1.1 Selecting an Object to Edit 1337
- 69.1.2 Editing an Object's Tags 1337
- 69.1.3 Changing an Object's Object Type Enumeration 1338
- 69.1.4 Changing an Object's Name 1340
- 69.1.5 Editing an Object's Description 1341
- 69.1.6 Specifying Whether an Object Can Be Created 1341
- 69.1.7 Specifying a Default User-Created Overlay 1341
- 69.1.8 Specifying the Platform 1342
- 69.1.9 Editing a Simulation Object's Country List 1342
- 69.1.10 Creating Icon Lists 1343
- 69.1.11 Undoing (Reverting) Changes to an Object 1345
- 69.1.12 Reverting an Inherited Object's Parameters 1345

69.2. Editing a Simulation Object's Visual Model 1346
- 69.2.1 Changing the 3D Model or Colorized Model 1347
- 69.2.2 Changing the 2D Icon Using a Font 1348
- 69.2.3 Changing the 2D Icon Using an Image 1349
- 69.2.4 Editing a Spot Report Model 1351
- 69.2.5 Adding a New Visual Model or Image 1351
- 69.2.6 Displaying Details about the Visual Model 1352
- 69.2.7 Using Render Settings 1354
- 69.2.8 Changing a Visual Model's Type 1356

69.2.9 Specifying Visual Markings 1357
69.2.10 Importing Visual Models 1359
69.2.11 Pruning Visual Models 1359
69.2.12 Exporting Visual Models 1359
69.2.13 Reloading Visual Models in the VR-Forces GUI 1360
69.3. Creating a New Simulation Object 1360
69.3.1 Creating a New Model by Copying an Existing Model 1361
69.3.2 Inheriting a New Model from an Existing Model 1361
69.4. Deleting an Object Model 1362
69.5. Creating a Screen Capture for the Create Object Palette 1362
69.6. Creating a Simulation Object Group 1362
69.6.1 Setting the Simulation Connection when Creating an SOG 1363
69.7. Editing Tactical Graphics 1363
69.7.1 Editing a Tactical Graphic's Menu Icon 1364
69.7.2 Setting the Default Values for Tactical Graphics Properties 1364
69.7.3 Specifying Tactical Graphic Properties that Can Be Edited in the GUI 1365
69.7.4 Editing the Visual Model for a Tactical Graphic 1366
69.8. Creating a Template 1369
69.9. Editing Systems in the Simulation Object Editor 1369
69.9.1 Creating a New System 1370
69.9.2 Editing Probability Tables 1371
69.9.3 Deleting a System 1374
69.9.4 Reverting a System 1374

69.1. Editing Generic Object Parameters

This section explains how to edit the parameters that are common to most objects. The procedures in this section apply to simulation objects, tactical graphics, cultural features, and other objects such as weather and contamination areas.

- By default, the SMSs provided with VR-Forces are read-only. It is strongly recommended that you not edit these SMSs. You should create copies or new SMSs that include them. For details, see "68.7. Creating New Simulation Model Sets" on page 1323.
- If one of the object parameters described in this chapter does not apply to a particular object type, it is either blank or uneditable in the Simulation Object Editor.

69.1.1 Selecting an Object to Edit

Most of the Simulation Object Editor procedures assume that you have selected an object in the object list to edit.

➤ **To select an object to edit:**

1. Optionally, in the object list, select the tags for the type of object you want to edit.
2. Select the object in the list.

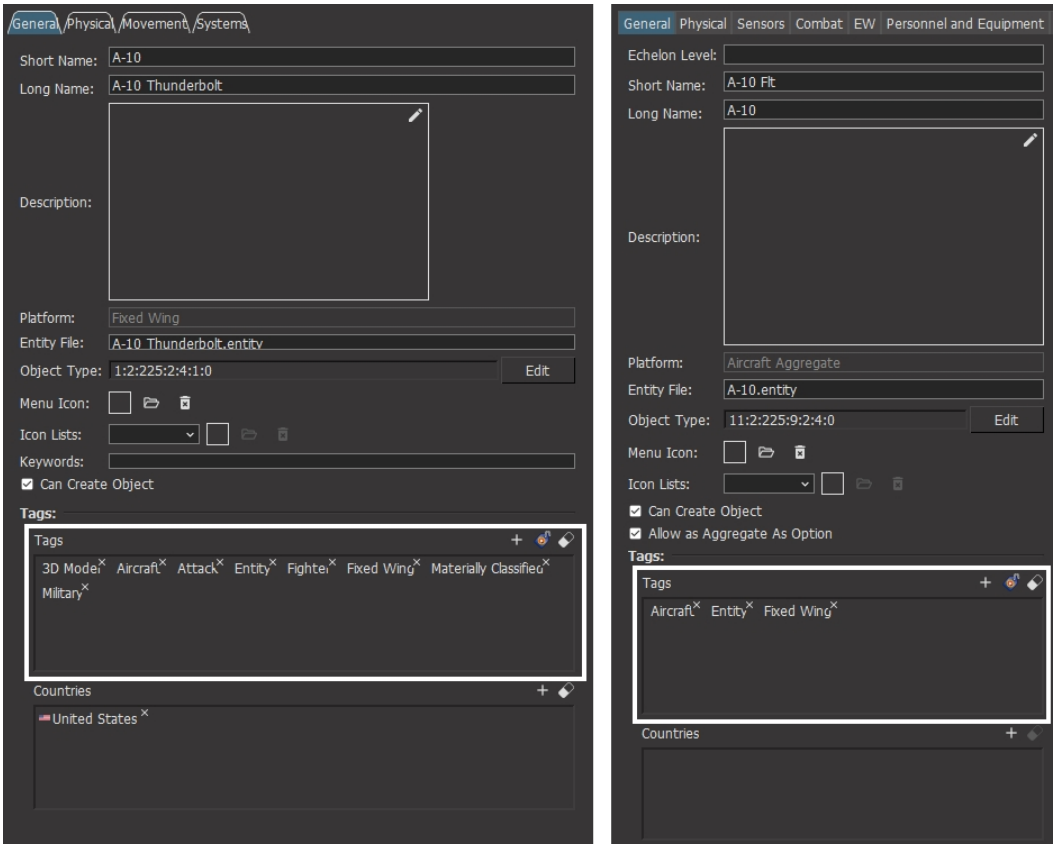
69.1.2 Editing an Object's Tags

In VR-Forces, objects are organized by tags on the Create menu and Create Object Palette. You can change the tags (or countries) assigned to an object. For details about adding new tags to the list, see "68.11.1 Adding a New Tag" on page 1327.

➤ **To assign a tag to an object:**

1. Select the object whose tags you want to edit.
2. Select the General tab (Figure 549).

Figure 549. General tab (entity and aggregate)



3. Click the Add/Remove button (+) above the Tags window. The Select Tags dialog box opens.
4. To select one additional tag and close the dialog box, click the tag.

   To add multiple tags, **Ctrl**+click the names of ones you want to add. You can finish and close the dialog box by either clicking the last tag or the Close button.

69.1.3 Changing an Object's Object Type Enumeration

The Simulation Object Editor lets you change an object's object type (DIS/RPR enumeration) and matching type. (For details about published and matching object types, see "Published Object Types and Matching Object Types" on page 1470.) You might want to or have to do this for any of these reasons:

- If you create a new model, you must specify an entity enumeration (Object Type). (The Simulation Object Editor automatically constructs a Matching Type whose Domain and Kind field match the Object Type and whose other fields are wild cards.)
- If you create a new model from an existing model, the Simulation Object Editor automatically increments the existing model's enumeration for use with the new model. However, you might want to use a different enumeration than the one the Simulation Object Editor generates. (The Simulation Object Editor automatically

creates a Matching Type that is the same as the Object Type.)

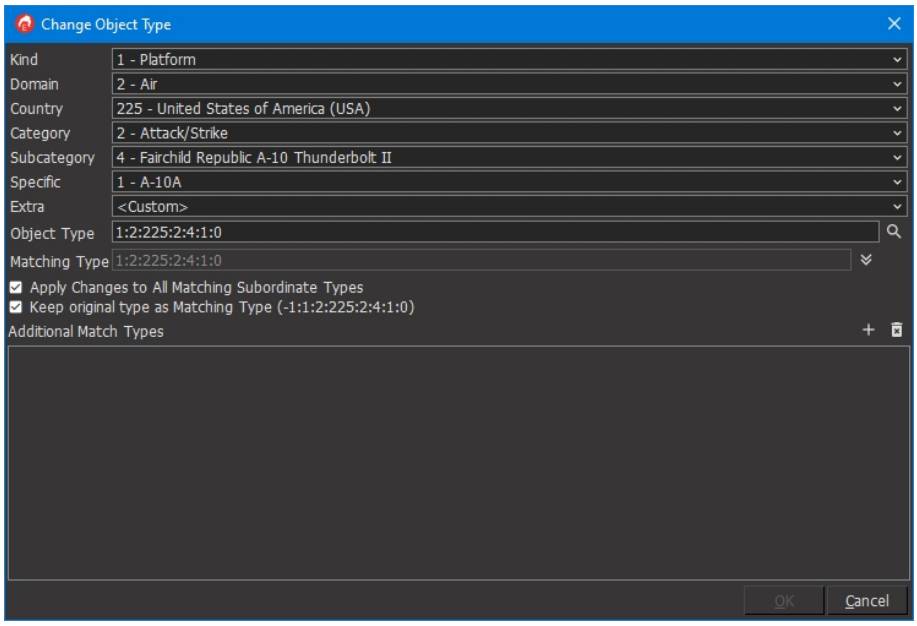
- You might want to change an existing object's enumeration.
- You might want to change an object's matching object type.

- The Entity Enumeration value displayed in the Simulation Object Editor is the 7-digit DIS or RPR FOM enumeration that VR-Forces uses to build the published object type. The matching object type is not displayed in the Simulation Object Editor, but you can specify it in the Change Enumeration dialog box as described in this procedure.
- You cannot use wildcards in an object type. They are permitted in the matching object type.

➤ **To change an object's enumeration:**

1. Select the object in the object list.
2. Select the General tab (Figure 549 on the previous page).
3. Click Edit next to the Object Type text box. The Change Object Type dialog box opens (Figure 550).

Figure 550. Change Object Type dialog box



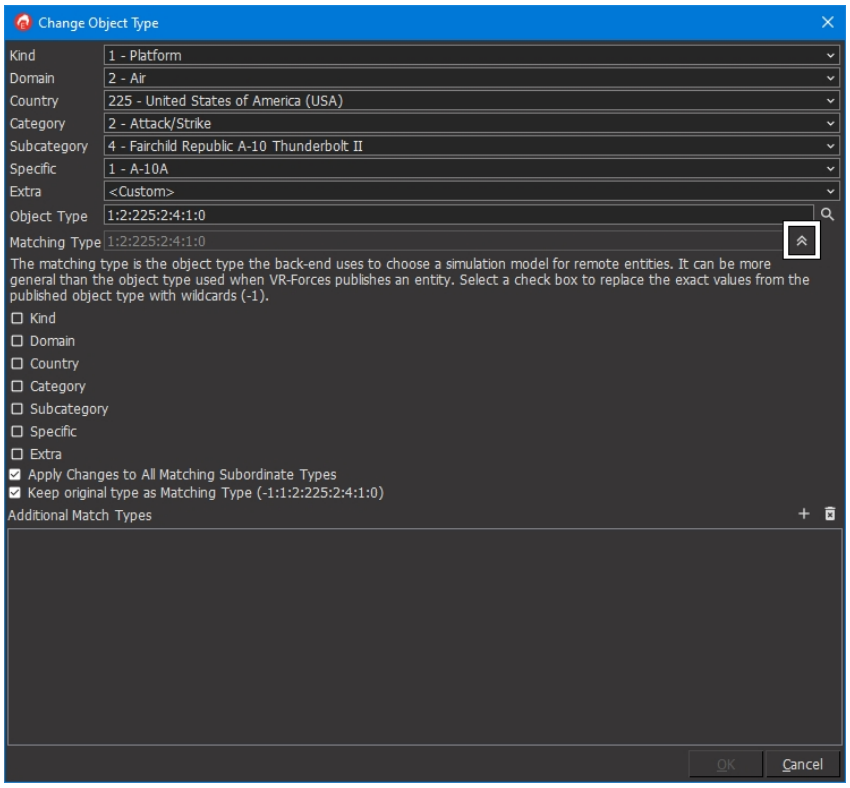
4. Edit the enumeration by selecting options from the lists for enumeration elements or enter a new enumeration in the Object Type text box.

   You can click the Search button (🔍) to open a dialog and search for the entity type by description.

5. Optionally, specify the Matching Type:

a. Click the double down arrow next to the Matching Type box. The dialog box expands to show the Advanced properties (Figure 551).

Figure 551. Change Object Type dialog box, Advanced



b. For each field in the matching type enumeration that you want to be a wild card (-1), select the appropriate check box.

6. Optionally, clear the Apply Changes to All Matching Subordinate Types check box. Usually you will want an enumeration change to affect all cases where this simulation object will be created. However, if there is some reason that you want an enumeration change to not affect simulation objects created as part of a unit, you can do that by clearing this check box.

   A more straightforward approach would be to create a new simulation object from the existing simulation object, rather than changing the enumeration.

7. Click OK. The enumeration changes.

69.1.4 Changing an Object's Name

The object name is the name listed on the object creation palettes and Create menu. The text placed on the screen next to a 2D icon is the marking text. The marking text is based on the Short Name, which is configured on the General tab.

➤ **To change an object's name:**

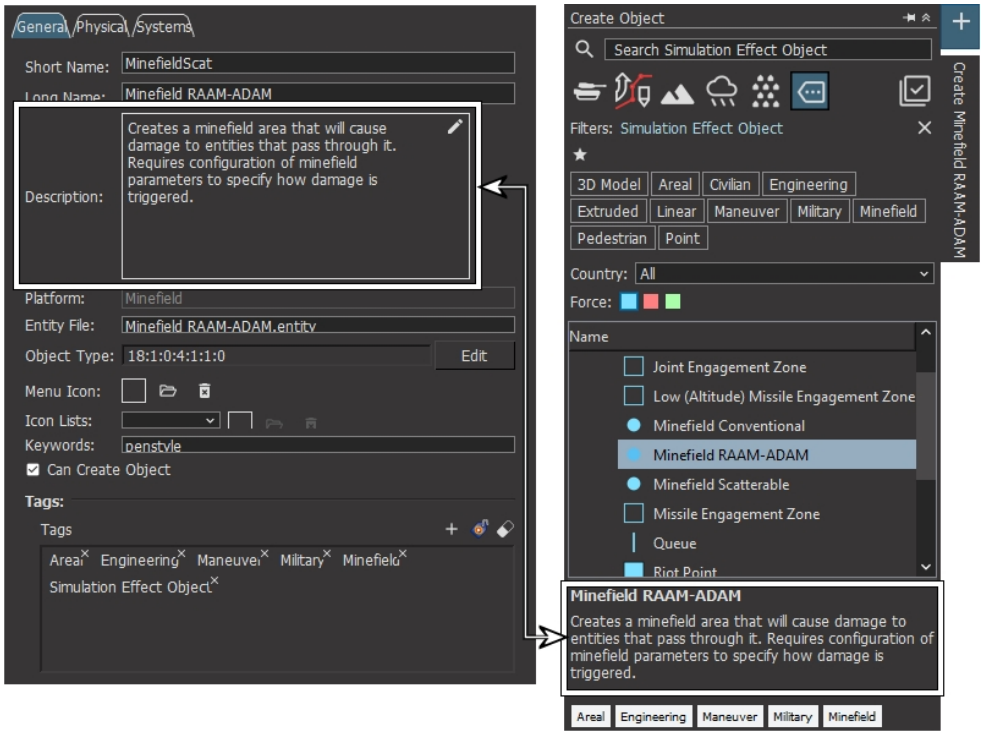
1. Select the object in the object list.
2. Select the General tab (Figure 549 on page 1338).

3. Enter a new name in the Long Name text box.

69.1.5 Editing an Object's Description

There is an optional field for the object's description. Descriptions display in the Create Object Palette in VR-Forces (Figure 552 below). It is useful for explaining objects whose purpose might not be straightforward before creating them.

Figure 552. Simulation object description



69.1.6 Specifying Whether an Object Can Be Created

You can control whether users can create a simulation object from the Create Object Palette. It may not make sense to be able to create some types of simulation objects, such as missiles, which are normally added to a scenario and removed dynamically.

➤ **To enable or disable creation of a simulation object from the Create Object Palette:**

1. Select the object in the object list.
2. Select the General tab.
3. Select or clear the Can Create Object check box (Figure 549 on page 1338).

69.1.7 Specifying a Default User-Created Overlay

When an object is created, it is placed on an overlay. By default, the object is placed on an overlay according to its type (aircraft, ground vehicle, human, tactical graphics, and so on). You can assign an additional user-created default overlay to objects. You can still assign the object to a different overlay when it is created. For details about overlays and

assigning objects to them, see "46.1. Creating and Editing Overlays" on page 1004 and "14.3.2 Specifying an Object's Properties Before You Create It (Click to Locate)" on page 384.

➤ **To specify a default overlay:**

1. Select the object in the objects list.
2. Select the General tab (Figure 549 on page 1338).
3. In the Keywords box, enter **overlay_***name* where *name* is the custom overlay.

69.1.8 Specifying the Platform

You cannot change the platform for existing simulation objects or new objects created from existing models. You can only specify the platform for new models that are not created from existing models. For details, see "69.3. Creating a New Simulation Object" on page 1360.

69.1.9 Editing a Simulation Object's Country List

The country code in a simulation object's object enumeration specifies the country in which the object is manufactured. It does not indicate where it is used, and many object types are used in multiple countries. When creating a scenario, it can be convenient if the Create Object Palette only lists the objects used by the country for which you are laying down forces. The Country list in the Simulation Object Editor allows you to specify the countries that use a vehicle. You can then filter the Create Object Palette by country.

You can also apply a country to humans and cultural features if they are specific to a country.

The Country list works exactly like the Tags list. For details about how to configure the list of countries, see "68.11. Editing the Tags List" on page 1327.

➤ **To assign a simulation object to a country of use:**

1. Select the object in the object list.
2. Select the General tab (Figure 549 on page 1338).
3. Click the Add button (+) above the Countries pane.
4. The Select Countries dialog box opens.
5. To select one country and close the dialog box, click the country name. To add multiple countries, **Ctrl**+click the names of the countries you want to add.

Clearing the Countries List

If you add a new model by copying another model, you may want to quickly clear the Countries list before selecting the countries for the new model. (You can clear the other lists the same way.)

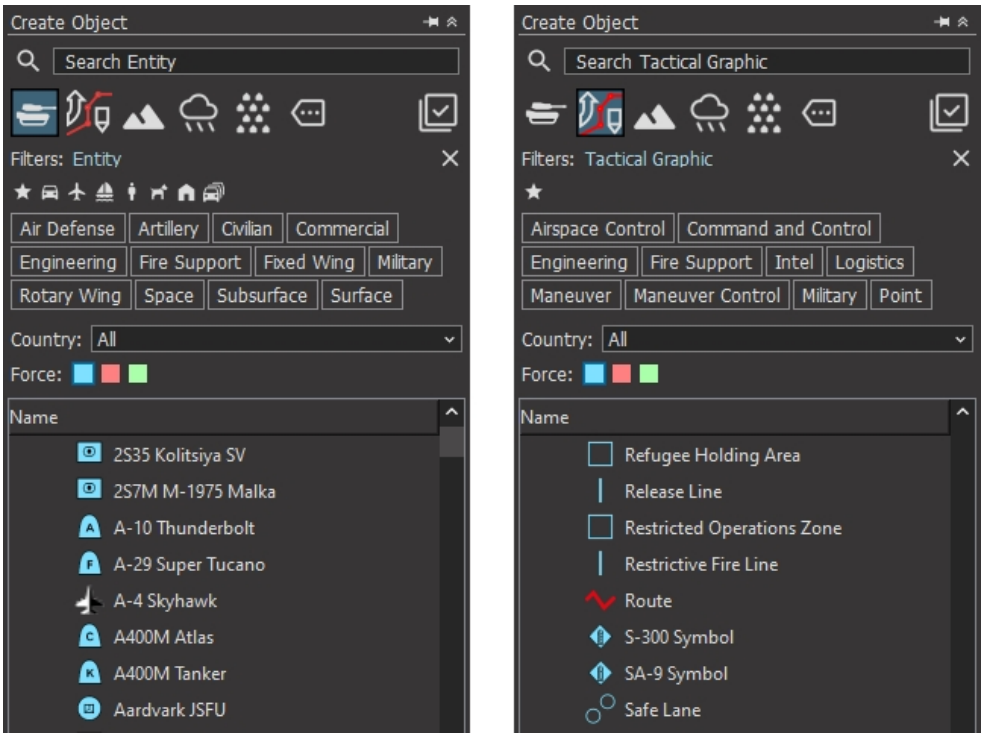
➤ To clear the Countries assigned to an object, on the General tab, click the Clear button (◈) above the Countries window.

69.1.10 Creating Icon Lists

By default, the object lists on the object creation palettes and the Objects List Panel show the 2D icon for each object. For entities, units, and some tactical graphics, this is the MIL-STD 2525B icon. For other tactical graphics, it is a simple graphic representation of the object, such as a line for linear graphics or a box for areal graphics. However, if you want to use images that are more specific to simulation object types, you can create custom icon lists that replace the default images. For example, instead of a straight line for a route, you might want to use a segmented line. Or instead of a 2D icon for an entity, you might want an image of the entity or a generic platform image.

Figure 553 illustrates object palettes with custom images (A-4 Skyhawk and Route).

Figure 553. Alternative icons



When you create a custom icon list, you create a list and then specify an image for each object that you want to change. If you do not specify an image for an object, it uses the default icon. You can create as many icon lists as you want. You specify the icon list to use on the Application Settings dialog box, GUI Settings page.

➤ **To create an icon list:**

1. Choose **Edit > Icon Lists**. The Icon Lists Editor opens.
2. Click Add. The Add List dialog box opens.
3. Enter a name for the new list.
4. Click OK.
5. In the Icon Lists Editor, click OK.

➤ **To change the name of an icon list:**

1. Choose **Edit > Icon Lists**. The Icon Lists Editor opens.
2. Select the icon list whose name you want to change.
3. Click Edit. The Edit List dialog box opens.
4. Enter a new name for the list.
5. Click OK.
6. In the Icon Lists Editor, click OK.

➤ **To remove an icon list:**

1. Choose **Edit > Icon Lists**. The Icon Lists Editor opens.
2. Select the icon list that you want to remove.
3. Click Remove.
4. Click OK.

Specifying Icons for an Icon List

After you create an icon list, you specify the images for each simulation object that you want to change the icon for.

➤ **To specify an image for an icon list:**

1. Select a simulation object in the objects list.
2. Select the General tab.
3. In the Icon Lists drop-down list, select the icon list that you want this object to use.
4. Click the Browse button (). The Select Menu Icon dialog box opens.
5. Select an image. It can be a PNG, SVG, JPG, or BMP file.
6. Click Open. The image displays next to the Browse button.
7. Repeat this procedure for each simulation object for which you want to specify an image.
8. Save the Simulation Model Set.

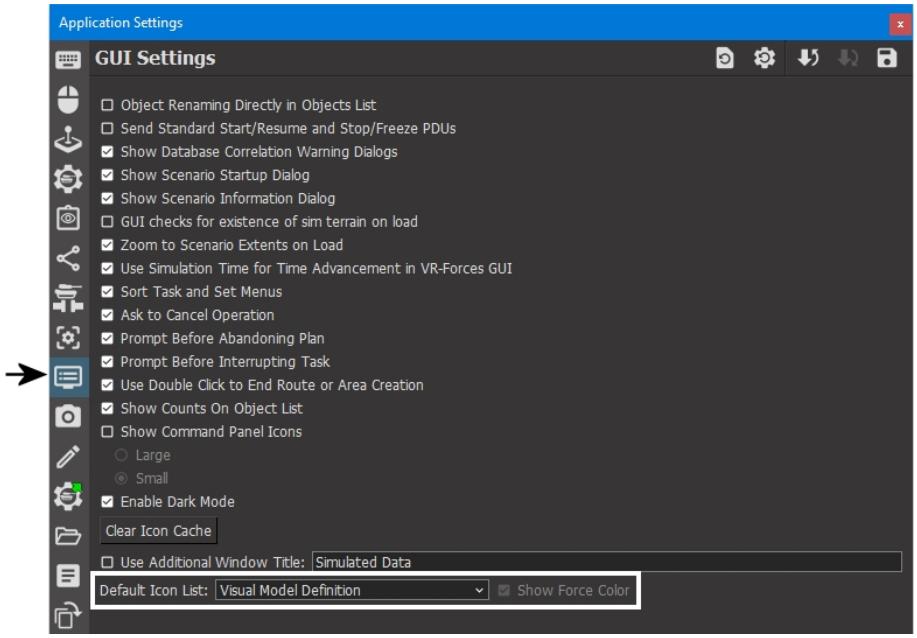
Using an Alternative Icon List

If you create an icon list, you can specify that VR-Forces use it instead of the default icon list.

➤ **To specify use of an alternative icon list:**

1. In VR-Forces, choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 554).

Figure 554. GUI Settings page



3. In the Default Icon List list, select the icon list that you want to use.
4. Optionally, select Show Force Color. This places a colored square behind the image to show its force.

69.1.11 Undoing (Reverting) Changes to an Object

If you have made changes to an object and have not yet saved the simulation model set, you can revert to the original simulation object configuration. You can revert an object at any time prior to saving the SMS, regardless of how many other objects you may have changed since you edited the simulation object you want to revert.

➤ **To undo changes to an object:**

1. Select the object in the object list.
2. Choose **Model > Revert Model**.

69.1.12 Reverting an Inherited Object's Parameters

You can create a new simulation object from another by inheritance, where you create a copy of the original (parent). See "69.3.2 Inheriting a New Model from an Existing Model" on page 1361. You can then make edits to those parameters. At any time, you can revert the parameters to the parent's parameters.

Please note, reverting a parameter resets it to the values that existed for the parent at the time you created the child. If the parameters changed on the parent after you created the child, reverting does not set the child parameters to the parent's new values.

➤ **To revert an inherited object's parameter to its parent's parameter:**

1. Select the object in the model list.
2. Next to the parameter that you want to revert, click the Revert to Parent button ( ).

69.2. Editing a Simulation Object's Visual Model

The Visual Model Editors dialog box allows you to configure all aspects of entity visualization, including the 2D icon and 3D model used. (For details, see the chapters in Section III, Visual Models, in *Adding Content to MAK ONE Applications Reference Manual*.) However, the Simulation Object Editor allows you to quickly and easily change the 3D model or 2D icon used to represent a simulation object. When you change a model in the Simulation Object Editor, the entity's model definition also changes.

You edit visual model parameters on the Visual Models Panel (Figure 555).

Figure 555. Visual Models Panel



The Visual Models Panel is a fully functional window that supports the same mouse-look and keyboard controls as the VR-Forces main window. If you change the observer view of the selected entity's visual model, the window maintains that offset as you select other entities. Press **Spacebar** to reset the view.

The panel has these buttons:

- Edit Visual Model. Automatically opens the correct editor based on the model set that is displayed.
- Edit Wake Model. Changes the wake settings for surface entities. (Hidden for other platforms.)
- Visual Effects. Lets you display how the model looks when trailing effects, shadows, and wakes are enabled.
- Switch Node States. Lets you display how the model looks when switch node states are set.
- Environment Settings. Lets you display what the model looks like at different times of day and with different wind and ocean states.

➤ **To change the visual model for a simulation object:**

1. In the object list, select the object whose model you want to change. The model for the currently selected model set is displayed in the Visual Model Panel.

   The visual model window responds to the same keyboard and mouse controls as the GUI.

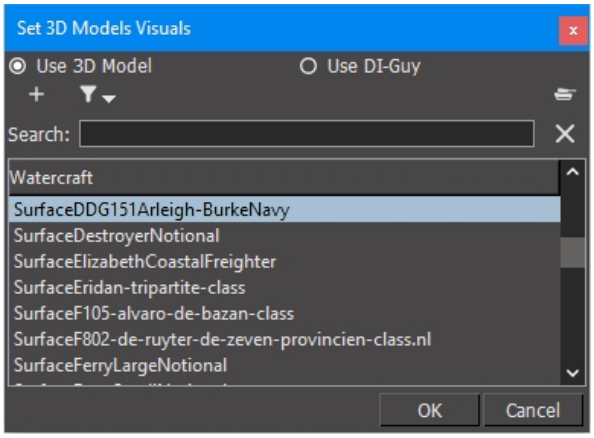
2. In the Model Set list in the Visual Model group box, select the model set that you want to edit for this simulation object.
3. Change the 3D model, 2D icon, or 2D image as described in the following sections.

69.2.1 Changing the 3D Model or Colorized Model

➤ **To change a 3D model or 3D colorized model:**

1. In the Visual Model panel, click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 556).
2. If the dialog box has options for 3D Models or DI-Guy, select Use 3D Model. The Modify Visuals dialog box lists available models (Figure 556). (You can add new models. For details, see "69.2.5 Adding a New Visual Model or Image" on page 1351.)

Figure 556. Modify Visuals dialog box (3D)



3. Select the model that you want to use. The new model displays in the panel.

> By default, the Modify Visuals dialog box tries to filter the list to show only the type of entity that you are editing. If you do not see the models you expect to see in the list, particularly if you know that the models are available in the *./SharedData/19/latest* directory, try changing the filter. For example, surface entities are of type Watercraft, while subsurface entities are of type Entity.

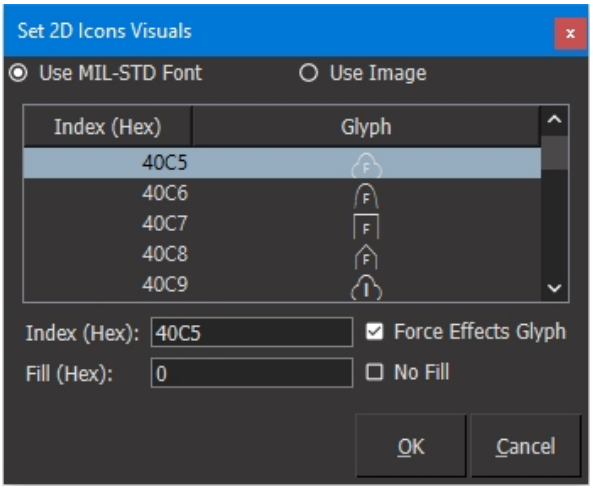
4. Click OK to close the Modify Visuals dialog box.

69.2.2 Changing the 2D Icon Using a Font

➤ **To change a 2D icon using the MIL-STD font:**

1. Select the simulation object whose icon you want to edit.
2. In the Visual Models Panel, in the Model Sets list, select 2D Icons.
3. Click the Edit button ( ). The Set 2D Icons Visuals dialog box opens (Figure 557).

Figure 557. Modify Visuals dialog box (2D)



4. Select the Use MIL-STD Font option. The dialog box lists index values and associated glyphs.
5. Select the icon glyph that you want to use. The new icon is displayed in the panel.
6. Optionally, change the parameters.
7. Click OK to close the dialog box.

Regenerating Spot Report Definitions

When you change the 2D icon for a simulation object, the spot report icon for that object does not change. You can change the icon manually, as described in "69.2.4 Editing a Spot Report Model" on page 1351, or you can automatically update spot report definitions.

➤ **To regenerate spot report definitions:**

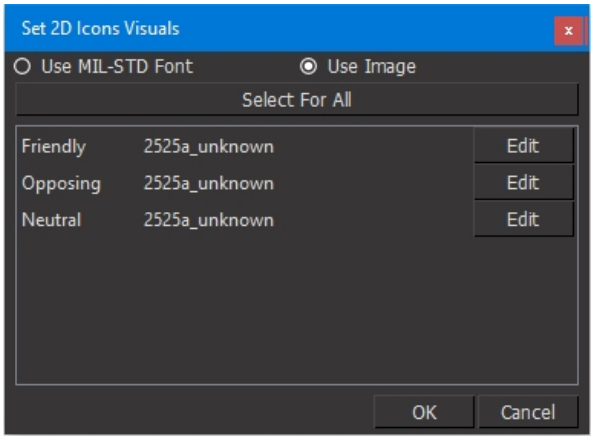
1. Choose **File > Regenerate Spot Report Definitions**. A prompt opens.
2. Click Yes.

69.2.3 Changing the 2D Icon Using an Image

➤ **To change a 2D icon using an image:**

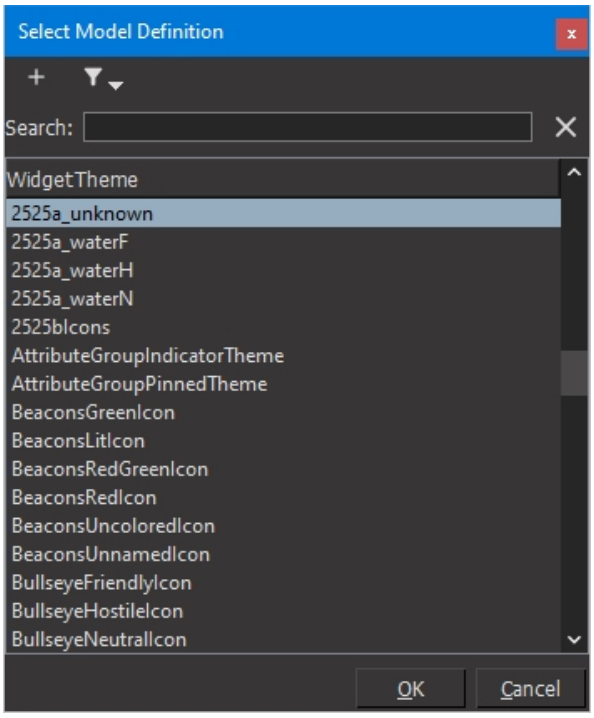
1. Select the simulation object whose icon you want to edit.
2. In the Model Sets list, select 2D Icons.
3. Click the Edit button (). The Set 2D Icons Visuals dialog box opens (Figure 557 above).
4. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set (Figure 558).

Figure 558. Modify Visuals dialog box (2D image)



5. Click the Edit button for the force whose icon you want to change or, to choose the same image for all forces, click Select for All. The Select Model Definition dialog box opens (Figure 559). It lists the available images. By default, the list is for MIL-STD 2525a icons. (You can add new images. For details, see "69.2.5 Adding a New Visual Model or Image" on the facing page.)

Figure 559. Select Model Definition dialog box



6. Select the image that you want to use. The new image is displayed in the panel.
7. Click OK.
8. Click OK to close the Modify Visuals dialog box.

69.2.4 Editing a Spot Report Model

You can edit the model that is used when a spot report is at the full knowledge level. The default configurations assign 2D icons to spot reports. In theory, you could assign some other type of model to a spot report. (The icons used at the detected, classified, and identified levels are not editable.)

➤ **To edit a spot report icon:**

1. Select the simulation object whose icon you want to edit.
2. In the Visual Models Panel, in the Model Sets list, select one of the spot report options.
3. Click the Edit button ( ). A dialog box appropriate to the model type opens.
4. Change the model, as described in "69.2.1 Changing the 3D Model or Colorized Model" on page 1347, "69.2.2 Changing the 2D Icon Using a Font" on page 1348, or "69.2.3 Changing the 2D Icon Using an Image" on page 1349.

69.2.5 Adding a New Visual Model or Image

When you change a model or image in the Simulation Object Editor, you are presented with a list of the currently configured model definitions or images. You can add additional models to the list. You can add new model definitions in the context of editing a simulation object or independently of editing a simulation object. If you edit the list in the context of editing a simulation object, you can also add images to the list.

Adding A Model Definition while Editing a Simulation Object

➤ **To add a new model or image to the visual models list while editing a simulation object:**

1. In the object list, select the object whose model you want to change. The model for the currently selected model set is displayed in the Visual Model group box.
2. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
3. For 3D models or 3D colorized models:
   a. Click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 556 on page 1348).
   b. Click the Add button ( ). The Model Definition dialog box opens. This is a standard file chooser dialog box.

   For 2D icons:
   a. Click the Edit button ( ). The Set 2D Icons Visuals dialog box opens (Figure 557 on page 1349).
   b. Select the Use Image option. The dialog box lists the forces that are configured for this simulation model set.
   c. Click the Edit button for the force to which you want to add an image. The Select Model Definition dialog box opens (Figure 559 on the previous page).

4. Select the file for the model that you want to add to the list. It must be a model or image type that VR-Forces supports. (This file becomes the value for the filename attribute of the model definition.)
5. Click OK. The Model Definition Name dialog box opens.
6. Enter a name for the model definition.
7. Click OK.
8. Click OK to close the Modify Visuals dialog box.

Adding Model Definitions Independently of Editing a Simulation Object

You can add model definitions in the Simulation Object Editor independently of any simulation object just as you can in the Visual Model Editors dialog box. The only attribute that you can set is the filename. However, in many cases that is the only attribute you need. If you need to specify other attributes, you must use the Visual Model Editors dialog box.

➤ **To add a model definition independently of editing a simulation object:**

1. Choose **Edit > Add Model Definition**. The Add Model Definition dialog box opens.
2. Click the Add button (+). The Model Definition dialog box opens. This is a standard file chooser dialog box.
3. Select the file for the model that you want to add to the list. It must be a model or image type that VR-Forces supports. (This file becomes the value for the filename attribute.)
4. Click OK. The Model Definition Name dialog box opens.
5. Enter a name for the model definition.
6. Click OK. The new model definition is added to the list.
7. Click OK to close the Add Model Definition dialog box.

69.2.6 Displaying Details about the Visual Model

The Visual Models Panel displays the model used for the selected simulation object. You can change how the model is displayed and show information about it:

- Bounding volumes. Display the bounding volume.
- Environment settings. Show the model at different times of day, different view magnification, different wind speed, and different water quality.
- Markers and marker labels. Displays labels for the markers for sensor and weapon positions. Also, show markers as 2D only.
- Visual effects. Show effects such as shadows, trailing effects, smoke, and flames.
- Model sets. Specifies which model set to use to display the entity.
- Switch node states. Displays the different switch nodes for the model, if applicable. If a model has articulated parts, you can enable them individually. They move at five times the minimum rotation or translation speed.

- View. Perspective view or view the model from the specified side (left, front, top, and so on.)

Changing the View of the Model

The default view for 3D models is a perspective view. You can also view a model from the front, back, sides, or top.

➤ To change the view of a model, in the Visual Models Panel, select an option from the View list.

Changing the Model Set

You can select the model set to use to display a visual model. In addition to the three primary model sets, you choose a spot report model set.

➤ To change the model set used to display a visual model, in the Visual Models Panel, select an option from the Model Sets list.

Displaying a Visual Model's Bounding Volume

You can display the bounding volume of a 3D model. You can edit the dimensions of the bounding volume on the Physical tab.

➤ To display a model's bounding volume, choose **Preferences > Show Bounding Volumes**.

Displaying Switch Node States

If a visual model has switch node states, you can display them. Switch nodes can represent damage states and articulated parts.

➤ **To display switch node states:**

1. Choose **View > Switch Nodes States** or click the Switch Nodes States button (). The Switch Nodes States dialog box opens.
2. To see a damage state, select an option from the Damage list.
3. To see articulated parts, select the check box for the state you want to display.
4. Click Close.

Displaying Markers and Marker Labels

You can display markers that show the location of support points for a model. You can display the markers as 2D or 3D and can also display text labels that identify them. You can edit the location of support points on the Physical tab.

➤ To toggle markers on and off, choose **Preferences > Show Markers**.

➤ To toggle marker labels on and off, choose **Preferences > Show Marker Labels**.

➤ To toggle markers between 2D and 3D, choose **Preferences > Show Markers as 2D**.

Displaying the Effects of Environment Settings

You can display how changing the following environment settings affects a visual model:

- Time of day
- View magnification
- Wind speed
- Wind direction
- Choppiness
- Ocean quality

Time of day affects all models. The other settings only affect surface entities.

➤ **To display the effects or environment settings for a model:**

1. Choose **View > Environment Settings** or click the Environment Settings button (). The Environment Settings dialog box opens.
2. Change the settings as desired.
3. Click Close.

Displaying Visual Effects for a Model

You can display the following visual effects for models:

- Dust clouds
- Contrails
- Tracks
- Flames
- Smoke
- Footprints
- Wakes
- Shadows
- Speed

If an effect is not valid for the selected model, the check box is not available.

➤ **To display visual effects for a model:**

1. Choose **View > Visual Effects** or click the Visual Effects button (). The Visual Effects dialog box opens.
2. Select the check box for each visual effect that you want to display.
3. To change the apparent speed of a moving model, adjust the Speed slider.
4. Click Close.

69.2.7 Using Render Settings

You can use the render settings while working with a simulation object.

➤ To access the Render Settings, choose **System > Render Settings**. The Render Settings window opens.

The render settings are:

- Wireframe Rendering Mode. Displays the polygons that make up the simulation object.
- Performance Statistics Overlay. Performance Statistics show information about the simulation object.
- Developer Debug Overlay. The SOE uses the ImGui API to create an overlay that developers can use to test changes to some of its visual features. It includes an overlay for testing ocean settings, shadow settings, and other visual features. It also includes an ImGui test user interface that demonstrates some of its features.

  This feature is provided "as is" for the use of interested developers. For complete information about how to use ImGui, please see documentation at https://github.com/ocornut/imgui and other internet resources.
- Channel Projection Resize Policy. The Projection Resize Policy determines how a channel behaves when you resize it. The options are:
  - Fixed. Keep the aspect ratio the same regardless of how the window is resized.
  - Horizontal. Change the horizontal field of view (FOV) to so that the ratio of the FOV angles matches the aspect ratio of the viewport.
  - Vertical. Change the vertical FOV so that the ratio of the FOV angles matches the aspect ratio of the viewport.

  You can change the projection resize policy for individual channels by editing the Projection Resize Policy attribute. You can also change the projection resize policy for all channels.
- Occluded Symbol Opacity.
- Crepuscular Rays. The SOE can simulate the effect of crepuscular rays (sunbeams) beaming through clouds.
- Depth of Field Autofocus.
- Droplets Scacle. You can change the size of water droplets for splash effects and sea spray.
- HBAO. Horizon-based ambient occlusion (HBAO) adds screen space ambient occlusion lighting effects to the scene. This improves the realism by approximating the amount of ambient light that impacts a scene, which provides more realistic lighting of three-dimensional objects. There are parameters that adjust how ambient occlusion appears:
  - HBAO Radius. The distance from occluding objects where the HBAO effect is visible. Higher values cause the effect to extend further from occluding objects.
  - HBAO Bias. The bias for thin items. At higher values, larger objects cause the most noticeable effect on surrounding objects. This can reduce polygon edge artifacts and artifacts caused by thin geometry.

  - HBAO Exponent. The intensity of the HBAO effect. Higher values create a more intense effect.
  - HBAO Scale Small AO. The intensity of the HBAO effect on objects that are close to each other. Higher values have a more intense effect on the small corner areas of objects and also objects that are close together.
  - HBAO Scale Large AO. The intensity of the HBAO effect on objects that are at the edges of the specified HBAO Radius. Higher values make the area for objects that are around the HBAO radius distance from each other show a more intense effect.
  - HBAO Depth Clamp Mode. The depth clamp mode to use. To Border uses a method that reduces the false occlusion of objects along the edge of the screen. To Edge uses a method that correctly applies the occlusion effect to objects along the edge of the screen.

- Occlusion Queries. If you are simulating on a terrain with many buildings, such as in a city, enabling occlusion queries can improve the frame rate. When they are enabled, the SOE draws draw bounding boxes around some objects and checks to see whether they are being rendered or are blocked visually. You can view the bounding boxes by enabling occlusion nodes and OC Draw Debug in the Developer Debug Overlay Terrain Patch Object Settings.
- Models Alpha to Coverage, Terrain Alpha to Coverage, and Alpha to Coverage Dither. Alpha to coverage is a way of handling objects with transparency that improves rendering of objects that have a sharp transition between opaque and transparent. Grass, trees, and fences are a few examples of when alpha to coverage is a good option. In addition to improving visualization of some objects, there is a slight performance improvement when alpha to coverage is enabled.

  If alpha to coverage is enabled, you can enable a dither shader that modifies the alpha with a noise pattern so that there are no hard edges between different alpha levels. The tradeoff for this setting is that it adds a small amount of noise as the object or camera moves.
- Debug Visuals. The SOE provides some shader debugging capabilities within its lighting shaders. The shaders are instrumented such that setting different flags (through uniforms) changes the behavior of the shader. Typically it changes fragment colors based on some parameter or parameters.

69.2.8 Changing a Visual Model's Type

The entity element definition hierarchy categorizes element definitions by type. These types associate a variety of parameters with the models in the category. Changing the visual type changes the 3D model for any simulation object that is using a generic model. If a simulation object has a specific model, changing the visual type does not affect it. However, it affects how the model is displayed in the SOE. Surface models are displayed with a background that lets you view wakes and other effects.

➤ **To change a visual model's type:**

1. In the object list, select the model whose type you want to change.
2. Choose **Model > Advanced Visual Model Editing**. The Advanced Visual Model Editing dialog box opens.
3. Select an option in the Visual Type list.

69.2.9 Specifying Visual Markings

VR-Forces uses a process called texture mapping to apply different surface textures to a simulation object's physical geometry. For example, you can use this technology to change the colors of a car, change the logo and design on aircraft, and change markings on a vehicle. The Simulation Object Editor allows you to see how different markings look on a vehicle, to apply specific markings to a vehicle, and to configure a vehicle to apply randomly assigned markings.

You can specify the visual markings that are applied to a simulation object when it is added to a scenario. Markings can be specific or random. A simulation object may have more than one type of visual marking. You can set each of them individually. The options are:

- Specific. Use one of the specific marking configured for the object.
- Random. Choose randomly from the markings available for the object. You can choose to be simply random or specify how to randomize the marking:
  - Weighted Random (20/80). Given a list of markings, there is an 80% chance that one of the markings in the top 20% of the list will be applied. There is a 20% chance that one of the other markings will be applied.
  - Weighted Random (Exponential). The markings towards the top of the list are more likely to be chosen then those later in the list.
- Variant. This is a possible pattern, such as country, for subvariant markings.
- Subvariant. Choose the manner in which the variant is displayed, for example, insignia.
- Unused. Do not show the option in VR-Forces.

For weighted random options, the order of the list is based on the order that markings are listed in the relevant metadata file. If you want the markings to have a different priority, you must edit the metadata file.

For example, if colors in the metadata file are listed in the order red, blue, green, yellow, brown and you select the weighted random approach, most cars will be red, because there is an 80% chance of selecting the color in the top 20% of the list. (In a list of five items, the first item in the list is the top 20%.) If you want most cars to be brown, you would need to edit the metadata file to put brown at the top. (For details about metadata files, please see "The Metafile for Markings" in the *Adding Content to MAK ONE Applications Reference Manual*.)

For details about how to change markings at run time, see "40.51. Markings" on page 888. For details about how to configure texture maps and their associated files, please see "Adding Dynamically Changeable Markings to Models" in the *Adding Content to MAK ONE Applications Reference Manual*.

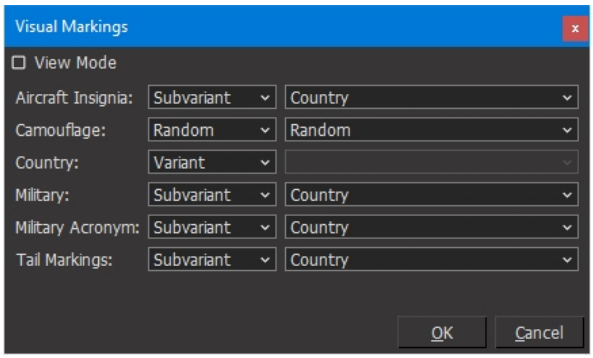
Viewing a Model's Visual Markings

The Simulation Object Editor lets you see how the available visual markings look on a simulation object.

➤ **To view a model's visual markings options:**

1. Select the simulation object.
2. In the Visual Model panel, click the Visual Markings button ( ). The Visual Markings dialog box opens (Figure 560). It has a list of options for each type of marking that the simulation object supports.

Figure 560. Visual Markings dialog box for the C-130 Hercules aircraft



3. Select specific options in the lists for each marking type that you want to view.
4. Click Cancel. (If you click OK, the options you have chosen are applied to the model.)

Setting Visual Markings

In the Simulation Object Editor, you can specify the visual markings that are applied to a simulation object when it is added to a scenario.

➤ **To set the visual markings for a simulation object:**

1. Select the simulation object.
2. In the Visual Model panel, click the Visual Markings button ( ). The Visual Markings dialog box opens (Figure 560).
3. For each type of marking:
   a. If you want to use a specific marking, select it from the list.
   b. If you want markings assigned randomly, select a random option in the list. For weighted random options, the markings at the top of your list are the most likely to be used.
4. Click OK.

69.2.10 Importing Visual Models

You can import visual models (element definitions, model definitions, and object type mappings) into an SMS. You might want to do this if you have edited an SMS and then decide that you need to revert to a previous version that you have saved in *./factory* or elsewhere. When you import visual models, the imported model definitions replace any matching definitions in the SMS. The import operation imports all leaf files in the selected directory.

➤ **To import visual models:**

1. Choose **File > Import Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory from which you want to import visual models.
4. Click OK.

69.2.11 Pruning Visual Models

Pruning visual models (element definitions, model definitions, and object type mappings) removes duplicate visual models from the set of leaf files that are loaded for an SMS. This improves the load time in the SOE and when you load a scenario.

Pruning only affects the SMS-specific definitions files. It does not touch any additional definition files that may have been saved from the Visual Model Editors dialog box.

➤ **To prune visual models:**

1. Choose **File > Prune Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory from which you want to prune visual models.
4. Click OK.

69.2.12 Exporting Visual Models

When you update an SMS, you can export the visual models (element definitions, model definitions, and object type mappings) to a directory other than the default directory in the SMS. This would typically be done for archival purposes, such as updating the files in *./factory*.

The SOE exports only the SMS-specific definitions files. If user-created leaf files were previously loaded, the definitions are not exported back to those leaf files. They are included in the SMS-specific files.

➤ **To export visual models:**

1. Choose **File > Export Visual Models**. A confirmation prompt opens.
2. Click Yes. A file chooser dialog box opens.
3. Select the directory to which you want to export visual models.
4. Click OK.

69.2.13 Reloading Visual Models in the VR-Forces GUI

If you update the visual models associated with simulation objects and you have a scenario open, you can refresh the models used in the scenario without shutting down VR-Forces.

➤ To refresh visual models, in the VR-Forces GUI choose **Settings > Reload Visuals**.

69.3. Creating a New Simulation Object

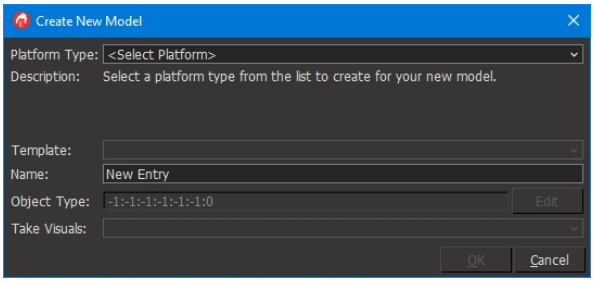
You can create a new simulation object by using an existing object as a template or by building a completely new object.

> The procedure for creating a new unit is largely the same as for creating a new entity. The main difference is that units do not have 3D models and, when you create a unit, you specify its subordinates.

➤ **To create a completely new object:**

1. Choose **Model > New Model**. The Create New Model dialog box opens (Figure 561).

Figure 561. Create New Entity dialog box



2. In the Platform Type list, select a platform type.
3. Optionally, select a template. (Templates are specific to a platform type. There might not be a template for some platforms.)
4. In the Name box, enter a name for the entity.
5. Optionally, for the Object Type, click Edit and specify the object type enumeration. You can do this after you create the entity if you want to. You can also just accept the object type supplied by VR-Forces.
6. In the Take Visuals list, select a specific model or Closest Match. If you select Closest Match, VR-Forces selects the entity model whose object type enumeration is closest to the one you specified.
7. Click OK. The Simulation Object Editor adds a new entry to the object list. The editor shows default values for all the fields.
8. Optionally, edit the enumeration.
9. Specify 2D and 3D visual models.

10. Specify appropriate values for the object parameters.
11. Enter a name to be used in the object's label (marking text) in the Short Name text box.
12. Specify a movement system.
13. Specify a damage model.
14. Optionally, add weapon systems.
15. Optionally, add sensors.
16. Save the simulation model set.

69.3.1 Creating a New Model by Copying an Existing Model

If you create a new simulation object model by copying an existing model, the new model starts with all of the behavioral characteristics and systems of the original one. It increments the object enumeration. You only need to name it and customize it for the new simulation object.

> You can use this procedure to create new units.

➤ **To create a new simulation object model as a copy of an existing model:**

1. In the object list, select the source object.
2. Choose **Model > New Model from Existing (Copy)**. The Create New Model dialog box opens.
3. Enter a name for the model.
4. Optionally, edit the enumeration.
5. Click OK. The new simulation object is added to the object list. If you did not edit the enumeration, the original enumeration is incremented. An *.entity* file is created based on the name.
6. Edit the parameters to provide the new simulation object the characteristics you want for it.
7. Save the simulation model set.

69.3.2 Inheriting a New Model from an Existing Model

If you create a new simulation object model by inheriting from an existing model, the new model (child) inherits its behavioral characteristics and systems from the original one. You can customize it as needed, but you can also revert it to the settings of the model from which it is inherited (parent).

You can use this procedure to create new units.

> If you want to undo any changes, you can revert them. See "69.1.12 Reverting an Inherited Object's Parameters" on page 1345.

➤ **To create a new simulation object model by inheriting from an existing model:**

1. In the object list, select the source object.
2. Choose **Model > New Model from Existing (Inherit)**. The Create New Model dialog box opens.
3. Enter a unique name for the model.
4. Optionally, edit the enumeration.
5. Click OK. The new simulation object is added to the object list. If you did not edit the enumeration, the original enumeration is incremented. An *.entity* file is created based on the name.
6. Edit the parameters to provide the new simulation object with the characteristics you want for it.
7. Save the simulation model set.

69.4. Deleting an Object Model

➤ **To delete an object model from a simulation model set:**

1. Select the object in the object list. (You can select multiple models using the **Shift** or **Ctrl** keys.)
2. Choose **Model > Delete Model** or press **Delete**.

69.5. Creating a Screen Capture for the Create Object Palette

In VR-Forces, the Create Object Palette shows a screen capture of the selected simulation object at the bottom of the palette. You can capture a new image to replace the default for an existing object or add a new one for a new object.

➤ **To capture an image for use in the Create Object Palette:**

1. Change the observer view in the Visual Model Panel as desired.
2. Choose **Edit > Create Screen Grab**.

69.6. Creating a Simulation Object Group

You create simulation object groups (SOGs) in the Simulation Object Editor. You specify the members of the SOG in a version of VR-Forces that is launched from the Simulation Object Editor. When the Simulation Object Editor launches VR-Forces, it needs to use a simulation connection. For details about setting the simulation connection, see "69.6.1 Setting the Simulation Connection when Creating an SOG" on the facing page.

Simulation object groups can include scripted tasks and sets. They become scenario-specific scripts when you add the SOG to a scenario.

Simulation object groups are saved with the extension *.sog*.

➤ **To create a simulation object group:**

1. Start the Simulation Object Editor.
2. Load the SMS you want to use.
3. Choose **Model > New Object Group**. The New Object Group dialog box opens.
4. Enter a name for the new SOG.
5. Click OK. The new group is added to the objects list and the Simulation Object Editor displays a simulation object group page.
6. Specify the tag, creation palette option, and other generic information as you would for any simulation object.
7. Optionally, change Creation Options and the altitude offset.
8. Click Launch Object Group Editor. You are prompted to save your changes to the SMS.
9. Click Yes. A limited version of VR-Forces opens. It has a grid terrain.
10. Use the Create Object Palette to add objects to the SOG as you would for a scenario. You can change to Stealth observer mode if you want to view the terrain in 3D. It is a flat terrain.
11. Choose **File > Save Object Group** to save the SOG and continue editing. Choose **File > Save Object Group and Exit** to save the SOG and close VR-Forces.

69.6.1 Setting the Simulation Connection when Creating an SOG

When you create an SOG, the Simulation Object Editor starts VR-Forces where you specify the members of the SOG. By default, it uses the DIS localhost connection for *EntityLevel.sms* and the HLA 1516 Evolved connection for the aggregate-level SMSs. You can change the connection if you want. (Aggregate-level scenarios must use HLA 1516 Evolved.)

➤ **To set the simulation connection when specifying the members of a simulation object group:**

1. In the Simulation Object Editor, choose **Edit > Object Group Connection**. The Choose Connection dialog box opens.
2. Select the connection you want to use.
3. Click OK.

69.7. Editing Tactical Graphics

You can add tactical graphics and edit existing tactical graphics just like you can add and edit simulation objects. You can create a new tactical graphic from an existing one or create a new tactical graphic using one of the basic graphic types: point, route, area, symbol, box, or circle.

When you edit a tactical graphic, you can change the default characteristics of that tactical graphic. When VR-Forces users create one of these graphics, they can edit its specific characteristics as desired.

The basic procedures for adding and editing tactical graphics are the same as for adding and editing simulation objects. This section describes procedures that are specific to tactical graphics.

69.7.1 Editing a Tactical Graphic's Menu Icon

Most objects on the Create menu and the various object creation palettes have icons associated with them. For simulation objects, the icon is used to represent them in the GUI. For tactical graphics, the icon illustrates the type of graphic. You can change the icon associated with a tactical graphic and add icons for newly created tactical graphics.

➤ To specify the palette icon for a tactical graphic, on the General tab, click the Browse button (🗁) for the Menu Icon line (Figure 549 on page 1338) and select the icon that you want to use.

➤ To remove the icon associated with a tactical graphic, click the Delete button (🗑) on the Menu Icon line.

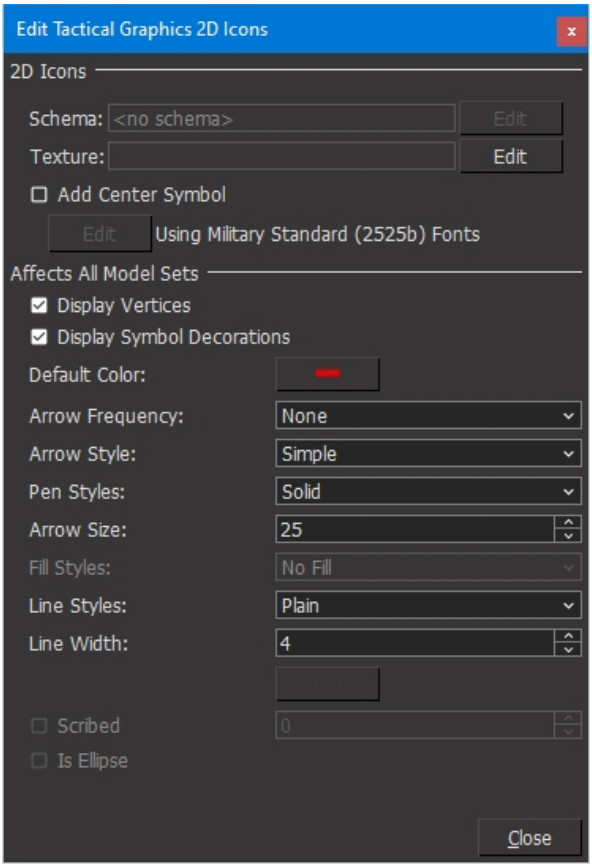
69.7.2 Setting the Default Values for Tactical Graphics Properties

When you select a tactical graphic, the Visual Model group box displays all of the properties that can be set for all of the different types of tactical graphics. Properties that do not apply to the selected tactical graphic are disabled. You can set default values for all of the properties that are applicable to the selected graphic. Setting a default value does not mean that the property will be editable in the VR-Forces GUI. The properties that are editable in the GUI are determined by keywords. For details, see "69.7.3 Specifying Tactical Graphic Properties that Can Be Edited in the GUI" on the facing page.

➤ **To set a default value for a tactical graphic property:**

1. In the Objects List, select the tactical graphic that you want to edit.
2. In the visuals panel, click the Edit button (✎). The Edit Tactical Graphics 2D Icons dialog box opens (Figure 562). The applicable properties are enabled.

Figure 562. Edit Tactical Graphics 2D Icons dialog box



3. Set the default values.

69.7.3 Specifying Tactical Graphic Properties that Can Be Edited in the GUI

When you create a tactical graphic in the GUI, you can edit its properties, such as line style, line width, color, and fill. The properties that you can edit in the GUI are determined by keywords that you specify in the Simulation Object Editor.

The keywords are:

- arrow. The user can change the object's arrow style.
- arrowsize. The user can change the object's arrow size.
- canorient. The user can change the object's orientation (heading).
- defaultlength=<num>. Specifies the starting length of the tactical graphic in meters.
- defaultwidth=<num>. Specifies the starting length of the tactical graphic in meters.
- extruded. The object's width and height can be extruded.
- fill. The user can change the object's fill.
- freehand. The object can be drawn freehand.

- initialaltitude=<num>. Specifies the starting altitude for the tactical graphic in meters.
- max=<num>. Specifies the maximum number of points allowed.
- min=<num>. Specifies the minimum number of points required to create the object.
- noaltitude. The user cannot set an altitude for the object.
- nostyle. Do not allow style editing.
- overlay=<name>. This is a way to define the default overlay for the tactical graphic.
- penstyle. The user can change line style for the object.

When you edit a tactical graphic in the GUI, VR-Forces builds an Edit Style dialog box with the properties specified by the keywords. If no editing is allowed, it removes the Edit icon from the Edit Object dialog box.

➤ To allow a tactical graphic's properties to be edited, on the General tab, enter the appropriate keyword in the Keywords text box. Separate each keyword with a semi-colon (;).

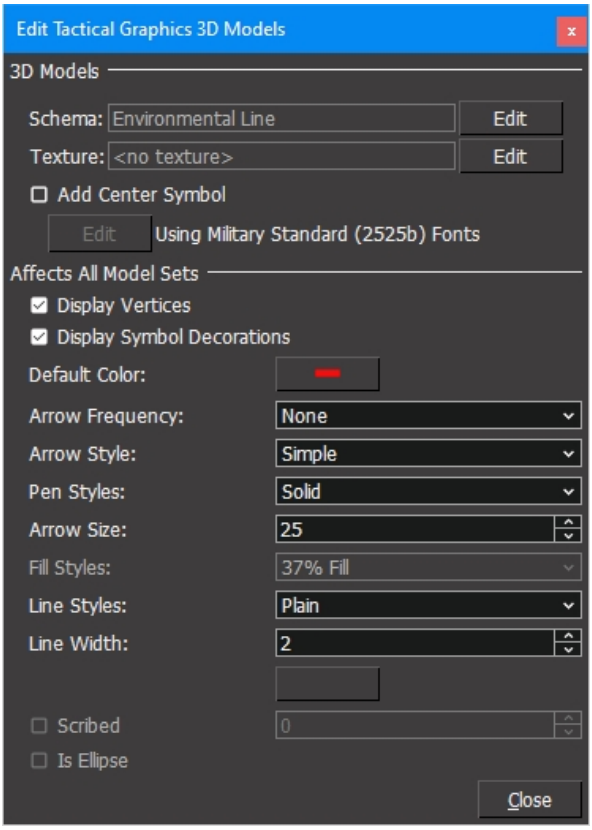
69.7.4 Editing the Visual Model for a Tactical Graphic

The visual model for a tactical graphic is based on a schema or on a texture. Schemas employ vector graphics and are best for basic lines, points, and areas. Textures apply raster images and are used primarily for specialized linear and areal graphics, such as fortified lines and minefields. (If you use a texture for a graphic, you cannot edit its style properties.)

➤ **To edit the visual model for a tactical graphic:**

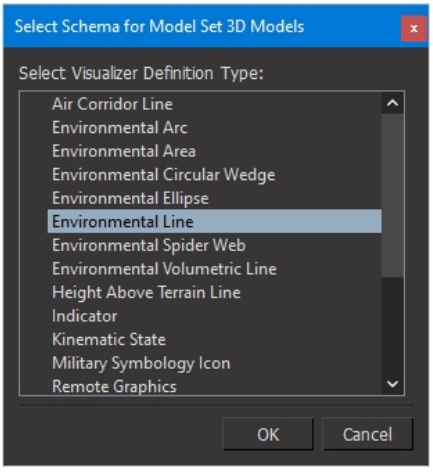
1. In the list of tactical graphics, select the one that you want to edit.
2. In the Visual Model panel, in the Model Sets list, select the model set that you want to edit.
3. Click the Edit button (). The Edit Tactical Graphics *model_set* dialog box opens (Figure 563), where *model_set* is the model set you selected.

Figure 563. Edit Tactical Graphics Models dialog box



4. On the Schema line, click the Edit button. The Select Schema for Model Set *model_set* dialog box opens (Figure 564).

Figure 564. Select Schema for Model Set dialog box



5. Select the schema that you want to use.
6. Click OK.

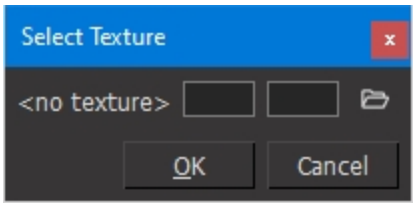
Editing the Texture for a Tactical Graphic

When you use a texture for a tactical graphic you specify a model definition and the height and width of the texture.

➤ **To specify a texture for a tactical graphic:**

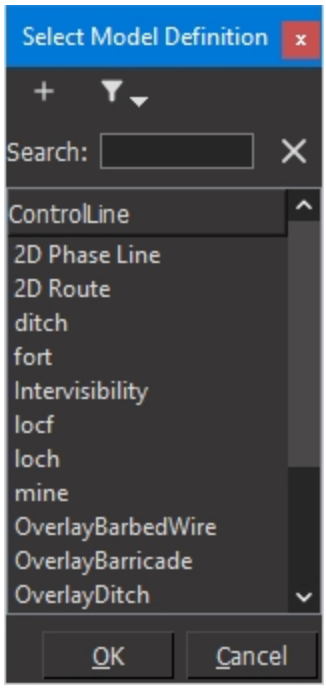
1. In the list of tactical graphics, select the one that you want to edit.
2. In the Visual Model group box, in the Model Sets list, select the model set that you want to edit.
3. Click the Edit button ( ). The Edit Tactical Graphics *model_set* dialog box opens (Figure 563), where *model_set* is the model set you selected.
4. On the Texture line, click Edit. The Select Texture dialog box opens (Figure 565).

Figure 565. Select Texture dialog box



5. Click the Browse button ( ). The Select Model Definition dialog box opens (Figure 566).

Figure 566. Select Model Definition dialog box



6. Select the model definition that you want to use for this texture.
7. Click OK.

8. In the Select Texture dialog box, specify the width and height of the texture, in pixels.
9. If this is an areal graphic, specify whether to use the texture for the outline of the area. If you do not use the texture, a regular line is used.
10. Click OK.

69.8. Creating a Template

When you create a new simulation object, you can select a template to base it on. You can create templates from existing simulation objects. Templates are meant as a starting point. They have standard posture, combat, and defense vales for that type of object. They do not include assemblies or health.

In many ways, basing a new model on a template is not much different from creating a new model from an existing model. However, when you create a template, you can edit it to be more generic than a particular simulation object, which makes it more broadly applicable. Templates are particularly useful for creating aggregate-level units because they are so much more complex than simulation objects in entity-level scenarios.

Templates are named with the format [Template] *object_name*, saved in the *./vrfSim/templates* folder of the SMS's directory, and included in the Template list of the Create New Model dialog box.

➤ **To create a template:**

1. In the objects list, select the simulation object that you want to use as a template.
2. Choose **Model > Make Template**. A template is created.
3. Optionally, select the template and edit it as you would any simulation object.

69.9. Editing Systems in the Simulation Object Editor

You can create and edit systems in the Simulation Object Editor. The support for different system types varies. Details of the use of the SOE to edit particular systems is described in the chapters devoted to those system types.

When you select a system, the SOE presents information and options on one or more tabs. All systems display the General tab, which can include:

- Name. System name.
- Description. A description of the system.
- Supported entity kinds. A list of the entity types that can use this system.
- Objects that use the system. This list is read-only. It updates when you add or remove a system from a simulation object.
- Available Simulation Object Parameters. A list of the parameters that you can edit for individual simulation objects. This field is read-only.
- Resource Defaults. Systems with resources, such as movement or weapon systems, have a default value for their resources.
- Tags. The tags that you want to apply to the system.

Most systems have a set of parameters that have default values. When you add a system to a simulation object, you can edit the parameters for that simulation object. The Attributes tab lists the system parameters for which you can set default values. Some systems have additional tabs for parameters, largely as a convenience to keep the window size manageable.

Weapon systems have a Targets tab that lets you configure hit probabilities, ammunition, and other targeting details. Damage systems have a Damage tab where you can configure how entities are damaged by different power types.

Munitions are on a separate tab in the Objects List Panel. They have a Detonation tab for configuring the power types supported by the munition.

The Damage Probabilities dialog box helps you understand the impact of different types of weapons on different types of entities.

69.9.1 Creating a New System

You can create a new system by copying an existing system or by basing it on a template. System templates represent a type of system that has a set of parameters that all systems of that type typically have. The choice between creating a new system by copying an existing one or using a template is a matter of estimating how much you must edit the system once you create it. For example, the parameter values for a 130 mm gun are probably very similar to those of a 125 mm gun, in which case you might want to simply copy the 125 mm gun. But they might be quite different for a 70 mm gun, in which case you might want to create the new system from a template.

Creating a New System from an Existing System

If you want to create a new system that is very similar to an existing system, you can copy the existing system to a new system and then edit the parameters.

➤ **To create a new system from an existing system:**

1. Select the Systems tab.
2. Select the system that you want to copy.
3. Choose **System > New System from Existing**. The New System from Existing dialog box opens.
4. In the System Name box, enter a name for the new system.
5. In the System Filename box, enter a filename for the new system or click the Browse button () and select a system definition file. If you select a system definition file, the new system will overwrite the existing system.
6. Click OK. The new system is added to the system list. Edit the system as desired.

Creating a New System from a System Template

If you want to create a system that has all the basic elements of a certain type of system, but which will not be very similar to an existing system, you can create it from a system template.

➤ **To create a new system from a system template:**

1. Choose **System > New System**. The New System from Template dialog box opens.
2. In the System Type list, select the type of system you want to create. If the type of system you want to create is not in the list, VR-Forces does not have templates for that system type.
3. In the System Template list, select the template that you want to use for the new system. The dialog box provides a brief description of the selected system type. It lists the systems that use the selected template and the simulation object types that support that system.
4. In the System Name box, enter a name for the new system.
5. In the System Filename box, accept the default system definition filename or enter the name you want to use.
6. Click OK. The new system is added to the system list.
7. Edit the parameters of the system.

69.9.2 Editing Probability Tables

Some systems, such as weapon systems and damage systems, use probability tables to determine the likelihood that a weapon will strike an entity, an entity will be damaged by a munition, and so on. These tables can be specified as a range list or a coefficient list.

A range list is a step function, a function that increases or decreases abruptly from one constant value to another. A coefficient list is a continuous function. Its values change according to a curve specified by a polynomial. You can specify the probabilities by editing a table. Once you create a table, you can edit it directly or graphically.

Figure 567 illustrates the following range list:

```
(range-list
  (range 1000.000
    (probability 0.9000)
  )
  (range 2000.000
    (probability 0.8000)
  )
  (range 3000.000
    (probability 0.7000)
  )
  (range 4000.000
    (probability 0.7000)
  )
)
```

Figure 567. Range list (step function)

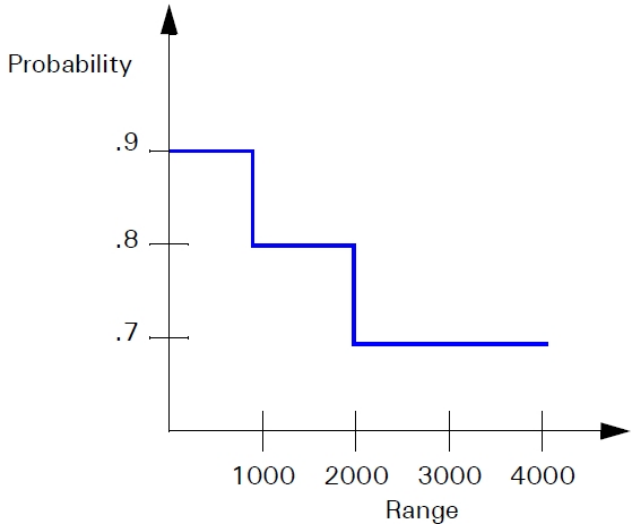
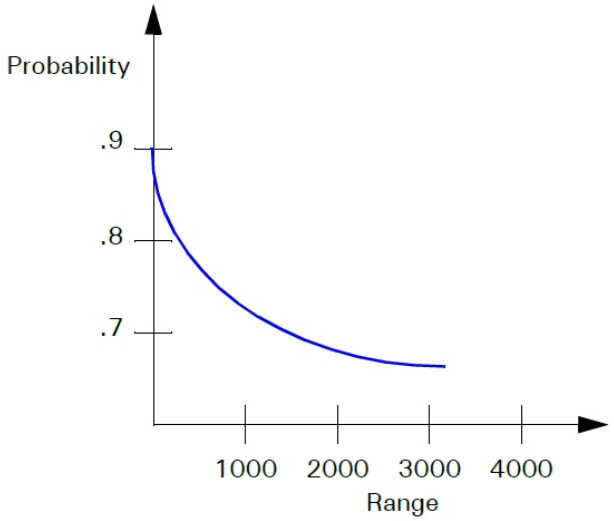


Figure 568 illustrates the curve specified by a coefficient list. The hit probability decreases along the curve rather than at specified steps. (If a coefficient list only has two points, it is graphed as a straight line, not a curve.)

Figure 568. Coefficient list polynomial curve



Editing a Range List Probability Table

Range lists are specific to a particular target type, damage power, or other system parameter. This procedure assumes that you have selected the system parameter for which you want to edit a range list.

➤ **To edit a range list table:**

1. To add a column to the table (that is, add a range value):
    a. Click the Add button (+) above the table. The New Range dialog box opens.
    b. Enter a range value.
    c. Click OK. A column is added to the table. It is automatically sorted to fit in the correct location relative to the other columns.
2. To edit a a range value:
    a. Select the column whose range you want to edit. (If you select a row, the column defaults to the first column.)
    b. Click the Edit button (✎) above the table. The Edit dialog box opens. (You can also double-click the range to open the Edit dialog box.)
    c. Enter a range value.
    d. Click OK. The range in the column heading is changed. The table is resorted if necessary.
3. To edit the probability for a given range:
    a. Select the probability value.
    b. Enter a probability as a whole number, for example, for 30%, enter 30.

Editing a Range List by Editing its Graph

When you edit a range list graph, you can change the range values and percentage for existing ranges. You cannot add new ranges to the table.

If a system has more than one parameter type, there is a graph for each type. You can edit each of them individually. (Depending on range settings, the graphs might overlap.)

Each graph has a different color. You can change the colors assigned to each graph.

➤ **To edit a range list graph:**

1. Move the mouse cursor over the graph. It changes to cross-hairs.
2. Hold down the mouse button above the horizontal or vertical portion of the graph that you want to edit.
3. Drag the line to the new value. The value in the corresponding cell in the table changes as you drag the line.

Editing a Coefficient List Probability Table

The columns in a coefficient list probability table have predetermined values: 1, r, r^2, r^3, and so on. You can set the value for each column.

➤ **To edit a coefficient list table:**

1. To add a column to the table (that is, add a coefficient value), click the Add button (+) above the table. A new column is added to the table.
2. To edit the probability for a given coefficient:

a. Select the probability value.

b. Enter a probability.

Changing the Color of a Probability Graph Line

Each system parameter that is being graphed is listed below the horizontal label for the graph. A color swatch next to the parameter name indicates the color used for that line in the graph.

➤ **To change the color used for a line in a probability graph:**

1. Click the label for the parameter whose color you want to change. A color picker dialog box opens.
2. Select the color you want to use.
3. Click OK.

Changing the Type of List for a Probability Table

You can change the type of probability table you use for a system parameter:

➤ To change from a range list to a coefficient list, click the Coefficient List button () above the table. Click Yes in the confirmation dialog box.

➤ To change from a coefficient list to a range list, click the Range List button () above the table. Click Yes in the confirmation dialog box.

69.9.3 Deleting a System

➤ **To delete a system:**

1. Select the system in the system list.
2. Choose **System > Delete System**. The system name is struck-through.
3. Save the SMS.

69.9.4 Reverting a System

If you edit a system and have not yet saved the SMS, you can revert all changes to the system since the SMS was last saved.

➤ **To revert changes to a system:**

1. Select the system in the system list.
2. Choose **System > Revert System**.
3. Save the SMS.



70. Editing Entities for Entity-Level Scenarios

This chapter explains how to use the Simulation Object Editor to edit individual objects in *EntityLevel.sms*. Many of the concepts and procedures also apply to units. For details about editing units, see "Chapter 71. Editing Units for Entity-Level Scenarios" on page 1405 and "Chapter 72. Editing Aggregate-Level Objects" on page 1417.

70.1. Editing Simulation Object Parameters 1377
- 70.1.1 Editing a Simulation Object's Bounding Volume (Size) 1377
- 70.1.2 Displaying a Simulation Object's Bounding Volume 1378
- 70.1.3 Specifying Support Points 1379
- 70.1.4 Excluding Entities from Batch Bounding Volume and Support Point Updates 1380
- 70.1.5 Editing Behavioral Parameters 1380
- 70.1.6 Adding Object Geometry to a Simulation Object 1380
- 70.1.7 Configuring Cultural Features To Be Embedded in the Terrain 1381
- 70.1.8 Ground Clamping Cultural Features 1382
- 70.1.9 Specifying Entity Properties that Can Be Edited in the GUI 1383
- 70.1.10 Editing a Lifeform's Visual Model and Animation 1383
- 70.1.11 Specifying a Default Appearance 1386

70.2. Editing Sensor Signatures 1387

70.3. Editing a Model's Articulated Parts 1387
- 70.3.1 Adding Articulated Parts 1387
- 70.3.2 Editing Articulated Parts 1388
- 70.3.3 Animating Articulated Parts 1389
- 70.3.4 Displaying Muzzle Flashes 1389
- 70.3.5 Removing Articulated Parts 1389

70.4. Editing a Simulation Object's Systems 1390
- 70.4.1 Adding a System 1391
- 70.4.2 Selecting a Damage Model 1392
- 70.4.3 Selecting a Movement Model 1392

70.4.4 Editing a System's Properties 1392
70.4.5 Removing a System 1392
70.4.6 Replacing a System 1393
70.5. Creating and Editing Munitions 1393
70.5.1 Creating a Simulated Munition 1393
70.5.2 Displaying Munition Detonations 1394
70.5.3 Creating a New Ammunition Type 1394
70.6. Configuring Wakes 1395
70.7. Creating Randomized Simulation Objects 1396
70.8. Configuring Scripted Entity Movement 1399
70.8.1 Configuring Ballistic Missile Movement 1399
70.8.2 Configuring Animated Movement 1401
70.8.3 Editing Movement File Configurations 1403

70.1. Editing Simulation Object Parameters

This section explains how to edit parameters that are only applicable to simulation objects and cultural features. (Cultural features are objects that you can add to a scenario to simulate human artifacts such as buildings, bridges, and so on. They are modeled as simulation objects because they exist in the physical world.)

70.1.1 Editing a Simulation Object's Bounding Volume (Size)

VR-Forces uses a simulation object's bounding volume to do intersection tests. The bounding volume is specified by the values in the Size/Supports section of the Physical tab. You can display a simulation object's bounding volume in the Visual Model panel. If you change size values, the bounding volume changes accordingly.

To set the bounding volume, you can:

- Edit the value manually.
- Calculate the bounding volume of a particular entity based on the model.
- Calculate the bounding volumes for all entities based on their models.

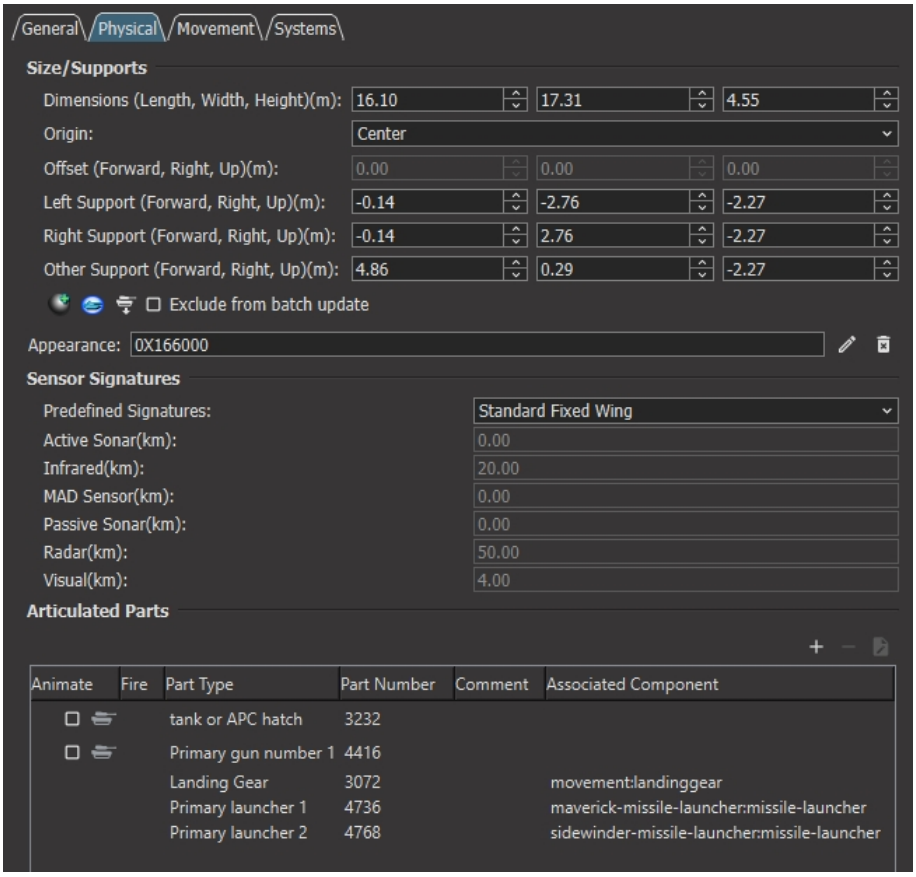


- The bounding volume is not the same thing as the selection box that is displayed around a simulation object when you select it. The selection box is just part of the GUI display. The bounding volume is used by the simulation engine.
- You can exclude individual entities from the global updates.
- VR-Forces calculates the bounding volume for units based on the locations and bounding volumes of the members of the unit.

➤ **To change an entity's bounding volume manually:**

1. Select the entity you want to edit in the object list.
2. Select the Physical tab (Figure 569).

Figure 569. Physical tab



3. Change any of the values in the Size/Supports section.

➤ To calculate the bounding volume of an entity from its model, click the Calculate Support Points and Bounding Volume from Displayed Model button ( ) or the Calculate Bounding Volume from Displayed Model button ( ).

➤ To calculate the bounding volume for all entities, choose **Model > Check Bounding Volume and Support Points**.

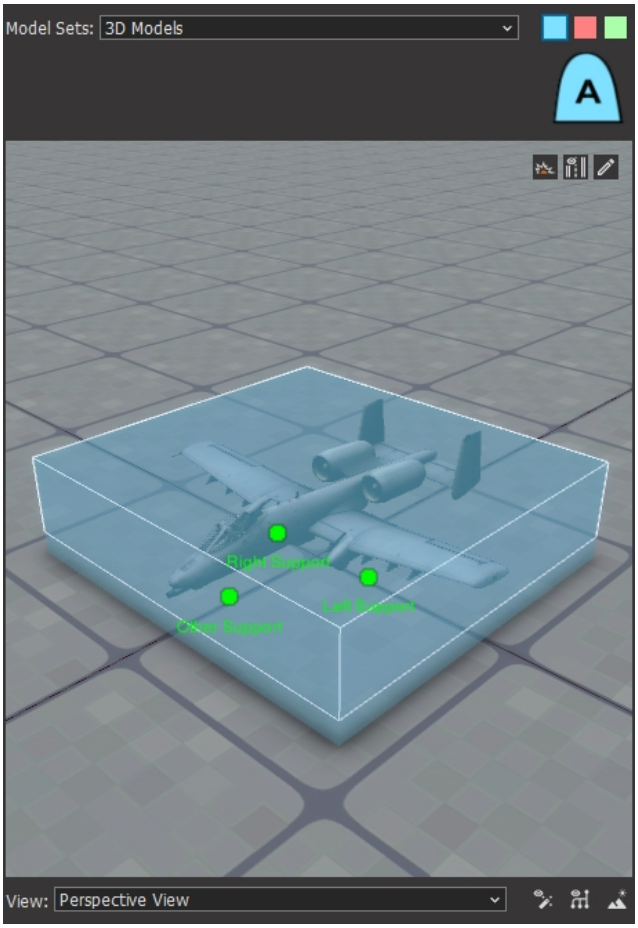
70.1.2 Displaying a Simulation Object's Bounding Volume

You must run VR-Forces at least once before you can display bounding volumes in the Simulation Object Editor.

➤ **To display a simulation object's bounding volume:**

1. Select the entity in the object list.
2. In the Visual Model Panel, from the Model Sets list, select 3D Models.
3. Choose **Preferences > Show Bounding Volumes**. The bounding volume displays (Figure 570).

Figure 570. Bounding volume



70.1.3 Specifying Support Points

Support points are the locations on a model that contact the ground. To specify support points, you can:

- Set them manually.
- Calculate them from the 3D model.
- Calculate the support points for all entities based on their models.

➤ **To specify support points manually:**

1. Select the entity you want to edit in the object list.
2. Select the Physical tab (Figure 569 on the previous page).
3. In the Size/Supports section, enter the desired values.

➤ To calculate support points based on the model, in the Size/Supports section, click the Calculate Support Points and Bounding Volume from Displayed Model button ( ) or the Calculate Support Points from Displayed Model button ( ).

➤ To calculate the support points for all entities, choose **Model > Check Bounding Volume and Support Points**.

70.1.4 Excluding Entities from Batch Bounding Volume and Support Point Updates

If you run the Compute Bounding Volume and Support Points command, it updates all entities. If you have specified the bounding volume or support points for an entity manually, this operation might override those settings. You can exclude individual entities from the global update process.

➤ **To exclude an entity from the global update process:**

1. Select the entity in the object list.
2. Select the Physical tab (Figure 569 on page 1378).
3. In the Size/Supports section, select the Exclude From Batch Update check box.

70.1.5 Editing Behavioral Parameters

The Simulation Object Editor lets you edit some of the parameters that affect simulation object behavior.

ⓘ The Sensor Signature parameters specify the distance at which other simulation objects can sense the simulation object you are editing. By contrast, the sensor systems on the simulation object control its ability to sense other simulation objects. For more information, see "70.4. Editing a Simulation Object's Systems" on page 1390.

➤ To edit simulation object parameters, edit any of the parameters in the Physical and Movement tabs. (The tabs reconfigure as you select different types of simulation objects.)

70.1.6 Adding Object Geometry to a Simulation Object

Object geometry is to a simulation object as terrain is to the earth. It allows embarked entities to move about on the entity on which they are embarked and it allows entities to move about on cultural features that are embedded in the terrain or embarked on other simulation objects. For example, adding object geometry to an aircraft carrier lets planes taxi on the deck of the carrier.

You can add object geometry to simulation objects in the Simulation Object Editor. Object geometry must be in OpenFlight, MEDF, or GDB format. OpenFlight geometry is converted to GDB and saved. For complete details about adding object geometry and generating navigation data, see "66.4. Generating Navigation Data for Entities" on page 1285.

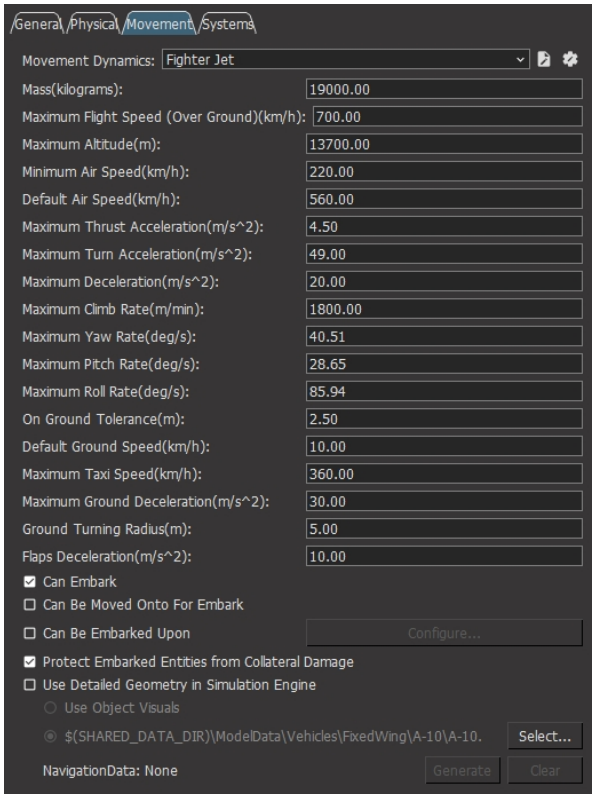
Ordinarily, you would specify an object geometry file for an entity. However, some entities have multiple main model visualizers in their element definitions. If an entity has multiple main models, VR-Forces can scan the element definition to identify all of the relevant object geometry files.

ⓘ When you edit an entity's embarkation points and slots, the model used is the visual model. This will not necessarily line up with the object geometry for that entity, if it has any.

➤ **To add object geometry to a simulation object:**

1. Select the simulation object in the Object List.
2. Select the Movement tab (Figure 571).

Figure 571. Movement tab for A-10 Thunderbolt



3. Select Use Detailed Geometry in Simulation Engine.
4. If the entity has multiple main models, select the Use Object Visuals option.
5. If it has just one main model, select the option that lists an medf file.
6. Optionally, to change the object geometry file:
    a. Click Select. The Select Object Geometry dialog box opens.
    b. Click the Browse button. The Object Geometry dialog box opens.
    c. Select the file that contains the object geometry for this entity.
    d. Click Open.
    e. Click OK.

70.1.7 Configuring Cultural Features To Be Embedded in the Terrain

If a cultural feature, such as a house, has object geometry, you can specify that it can be embedded in the terrain. When you place such an object into a navigation area, VR-Forces automatically generates navigation data for that object. This allows entities to walk

around and inside the building (if it has interior geometry) as if it were part of the terrain. For more information, see "14.16.1 Embedding Cultural Features in the Terrain" on page 401.

> You do not need to generate navigation data in the Simulation Object Editor for cultural features that you want to embed in the terrain. The navigation data is generated when you place the cultural feature in a scenario. Generating navigation data for an object in the Simulation Object Editor is for when you want to embark an entity onto that object.

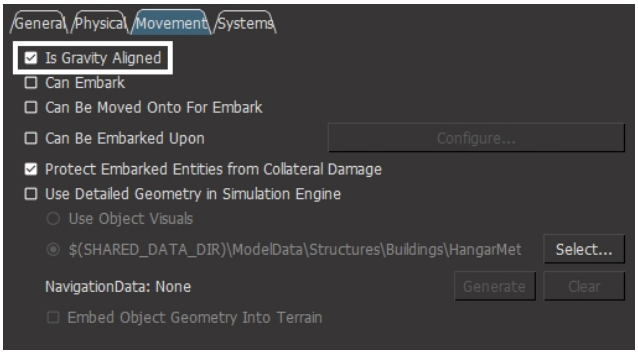
➤ **To allow a cultural feature to be embedded in the terrain:**

1. Select the cultural feature in the Objects List.
2. Select the Movement tab (Figure 572 below).
3. Select Use Detailed Geometry in Simulation Engine.
4. Click Select. The Select Object Geometry dialog box opens.
5. Click the Browse button. The Object Geometry dialog box opens.
6. Select the file that contains the object geometry for this entity.
7. Click Open.
8. Click OK.
9. Select Embed Object Geometry Into Terrain.

70.1.8 Ground Clamping Cultural Features

You can specify that cultural features be aligned to the terrain (ground clamped). If they are not aligned to the terrain, they might end up sticking out at odd angles.

Figure 572. Is Gravity Aligned check box



➤ To align cultural features to the terrain, on the Movement tab, select the Is Gravity Aligned check box (Figure 572).

70.1.9 Specifying Entity Properties that Can Be Edited in the GUI

When you create a simulation object in the GUI, you can edit its properties, such as pitch. The properties that you can edit in the GUI are determined by keywords that you specify in the Simulation Object Editor.

The keywords are:

- canbeairborne. The object can be in the air. This is used for things such as fire and smoke; it is not needed for aircraft.
- cansetpitch. The user can change the object's pitch.
- no-orbat. This object cannot be added to the Order of Battle panel for future creation through the Order of Battle panel.
- nocopy. Once placed, the entity cannot be copied.
- noedit. Once placed, the user cannot edit the object.

➤ To allow a simulation object's properties to be edited, on the General tab, enter the appropriate keyword in the Keywords text box. Separate each keyword with a semi-colon (;).

70.1.10 Editing a Lifeform's Visual Model and Animation

VR-Forces can specify the character, appearance, hand item, head, and animation values for DI-Guy models used in the 3D view and visualization applications such as VR-Vantage. A model's character determines the kinds of movement it can make. The appearance determines what it looks like. The animation specifies the movements that the character makes. In the VR-Forces GUI, you can specify the animation with the DI-Guy Animation task. You can set the appearance with the DI-Guy Appearance set data request. This feature is valid only for lifeforms (the Human and Animal tags).

➤ **To specify the default DI-Guy characteristics for a lifeform:**

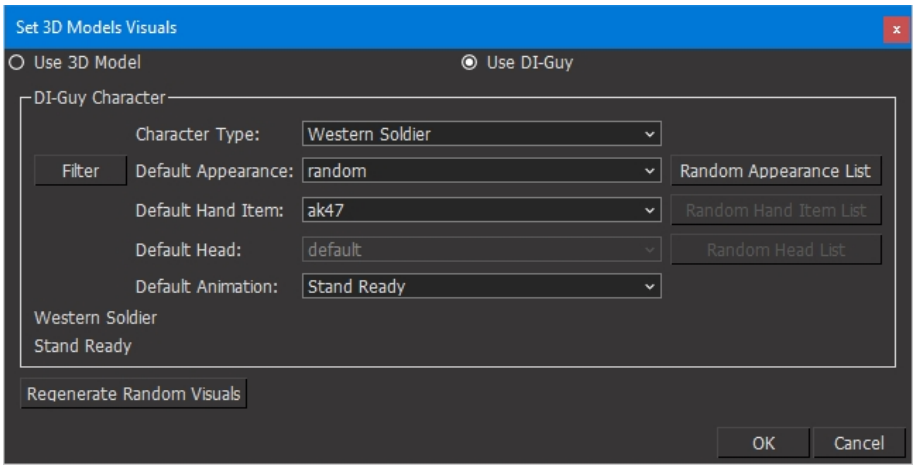
1. Select the Human or Animal tag.
2. Select a lifeform in the object list.
3. In the Visual Model Panel (Figure 573), on the Model Sets list, if 3D Models is not selected, select it.

Figure 573. Visual Model Panel



4. In the 3D display, click the Edit button ( ). The Set 3D Models Visuals dialog box opens (Figure 574). The Use DI-Guy option should be selected. If it is not selected, select it.

Figure 574. DI-Guy Character group box



5. Optionally, filter the options available in the lists:
    a. Click Filter. The DI-Guy Filters dialog box opens.
    b. Select an option for each characteristic that you want to filter on.
    c. Click OK.
6. Select the DI-Guy character you want to use from the Character Type list.
7. Select the default appearance from the Default Appearance list. One of the default options is to use a random appearance.
8. If you selected random appearance, optionally, specify the list of appearances to choose from. For details, see "Configuring the Random DI-Guy Characteristics Lists" below.
9. Optionally, in the Default Hand Item list, select a hand item. If you choose random, click Random Hand Item List to specify the hand items that can be chosen.
10. Optionally, in the Default Head list, select a head. If you select random, you can specify the list of heads to choose from. If you choose random, click Random Head List to specify the heads that can be chosen.
11. Select the default animation from the Default Animation list. The default animation is the animation the entity uses if it is not tasked and if it has a posture and speed that is appropriate to the animation.

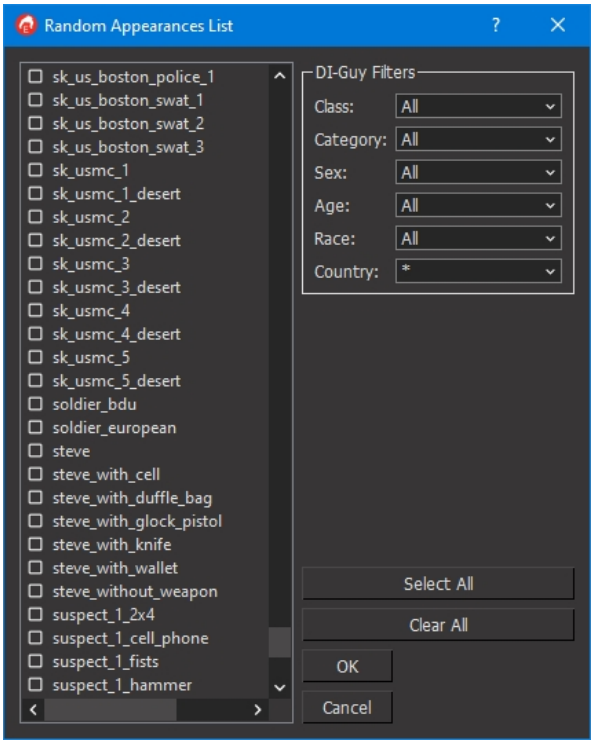
Configuring the Random DI-Guy Characteristics Lists

When you configure the visual model for a DI-Guy, you can specify that a lifeform use a random appearance, hand item, head, or some combination of these. When VR-Forces creates a lifeform, it chooses from the random characteristic lists. By default, a random list has all options available for the chosen character. However, you can specify a subset of the available characteristics to use.

➤ **To configure the DI-Guy characteristics lists:**

1. Select the Human tag (or Animal tag).
2. Select a lifeform in the object list.
3. In the Visual Model Panel (Figure 573), on the Model Sets list, if 3D Models is not selected, select it.
4. In the 3D display, click the Edit button (). The Set 3D Models Visuals dialog box opens (Figure 574). The Use DI-Guy option should be selected. If it is not selected, select it.
5. Click the random list button for the characteristic you want to configure. A dialog box opens (Figure 575). The options available for the character type are listed.
6. Optionally, filter the list by selecting options in the filter lists.

Figure 575. Random Appearances List dialog box



7. Select the options that you want to include in the random list.
8. Clear any options that you do not want to include in the random list.
9. Click OK.
10. Click OK to close the Modify Visuals dialog box.

70.1.11 Specifying a Default Appearance

You can specify the appearance bits for a simulation object that should be used when the object is first created in a scenario. For example, you might want a launcher to default to raised and operational.

➤ **To specify the simulation object's default appearance:**

1. Select the entity you want to edit in the object list.
2. Select the Physical tab (Figure 569 on page 1378).
3. Next to the Appearance box, click the Edit button (). The Set Appearance dialog box opens.
4. Set the appearance values as desired.
5. Click OK.

The controller also affects some appearance bits. For example, the Power Plant On might be set to No in the Simulation Object Editor, but the controller sets to Yes when creating the entity.

70.2. Editing Sensor Signatures

Sensor signatures specify the distance at which other entity sensors can locate a simulation object. VR-Forces includes a set of predefined sensor signatures for each of the sensor domains supported. You can edit the predefined sensor signatures in *./data/simulationModelSets/EntityLevel/extra/sensor-signatures.csv*.

> The VR-Forces simulation engine does not read *sensor-signatures.csv*. The file is only used to set predefined sensor signatures in the Simulation Object Editor. If you make change to the predefined sensor signatures, you must update objects in the Simulation Object Editor to put the changes into effect.

➤ **To specify sensor signatures:**

1. Select the simulation object.
2. Select the Physical tab (Figure 569 on page 1378).
3. In the Sensor Signatures section, specify a value in each domain in which the simulation object can be sensed or select a set of predefined signatures in the Predefined Signatures list.

70.3. Editing a Model's Articulated Parts

The Simulation Object Editor allows you to manage a simulation objects' articulated parts. This applies to articulated parts that are defined for the model, such as propellers, rotors, flaps, wheels, landing gear, tracks, radar, turrets, gun barrels, missile launchers, doors, hatches, and so on.

The default configuration for articulated parts comes from sim components and entity visual models. All parts from these sources are combined into a single list. These default settings can be overridden or have other settings added at the entity level using the Simulation Object Editor. The overrides to part settings are stored in the *.entity* file for the entity.

70.3.1 Adding Articulated Parts

You can add articulated parts to a simulation object. If they are associated with a part on the model that does not articulate visually, they can still have articulation information.

➤ **To add an articulated part:**

1. Select the simulation object.
2. Select the Physical tab (Figure 569 on page 1378).
3. In the Articulated Parts section, click Add Part (+). The Add Articulated Part dialog box opens.
4. You can hover the cursor over an available part to see the source file information.
5. Select the part to add and then click OK. The part is added to the list of articulated parts.

6. You can edit the part as needed; see "70.3.2 Editing Articulated Parts" below.
7. Save the simulation model set.

70.3.2 Editing Articulated Parts

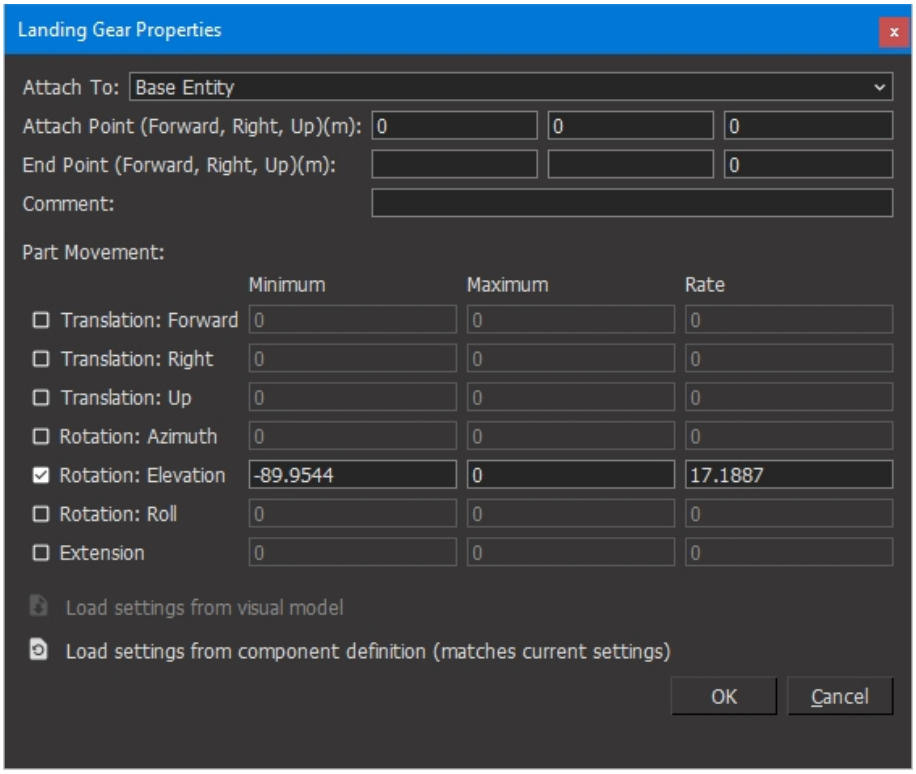
You can make changes to articulated parts on the simulation object. Your edits override any other settings. The articulated parts on the simulation object come from the system definition files, the model definition, and any articulated parts added to the simulation object.

Hover the cursor over a part in the list to see the source file information.

➤ **To edit an articulated part:**

1. Select the simulation object.
2. Select the Physical tab.
3. In the Articulated Parts section, select the articulated part you want to edit.
4. Click Edit Part Properties ( ). The properties dialog box opens (Figure 576).

Figure 576. Properties dialog box for articulated parts



5. Make changes as needed.
6. Click OK.
7. Save the simulation model set.

70.3.3 Animating Articulated Parts

You can see the animations of articulated parts that are associated with a part of the model that can be animated.

➤ **To animate an articulated part:**

1. Select the simulation object.
2. In the Visual Model Panel, from the Model Sets list, select 3D Models.
3. Select the Physical tab.
4. In the Articulated Parts section, select the Animate check box for the articulated part. The part animates in the Visual Model Panel.

70.3.4 Displaying Muzzle Flashes

You can preview the particle system configured for a weapon system to check whether it looks correct. The particle system displays at the point where it is defined. Munition types are configured with a specific particle system.

Select the part type to see the placement of the points. These are defined in the entity element, entity type map, and so on.

- The preview does not animate the articulated part in any way. It only shows the associated particle system.
- This does not work for lifeforms equipped with weapons.

➤ **To display a muzzle flash:**

1. Select the simulation object.
2. Select the Physical tab (Figure 569 on page 1378).
3. In the Articulated Parts section, click Fire ( ) for the weapon that you want to see. The particle system displays.

70.3.5 Removing Articulated Parts

You can remove articulated parts that you added to a simulation object. You cannot remove articulated parts that are part of the model.

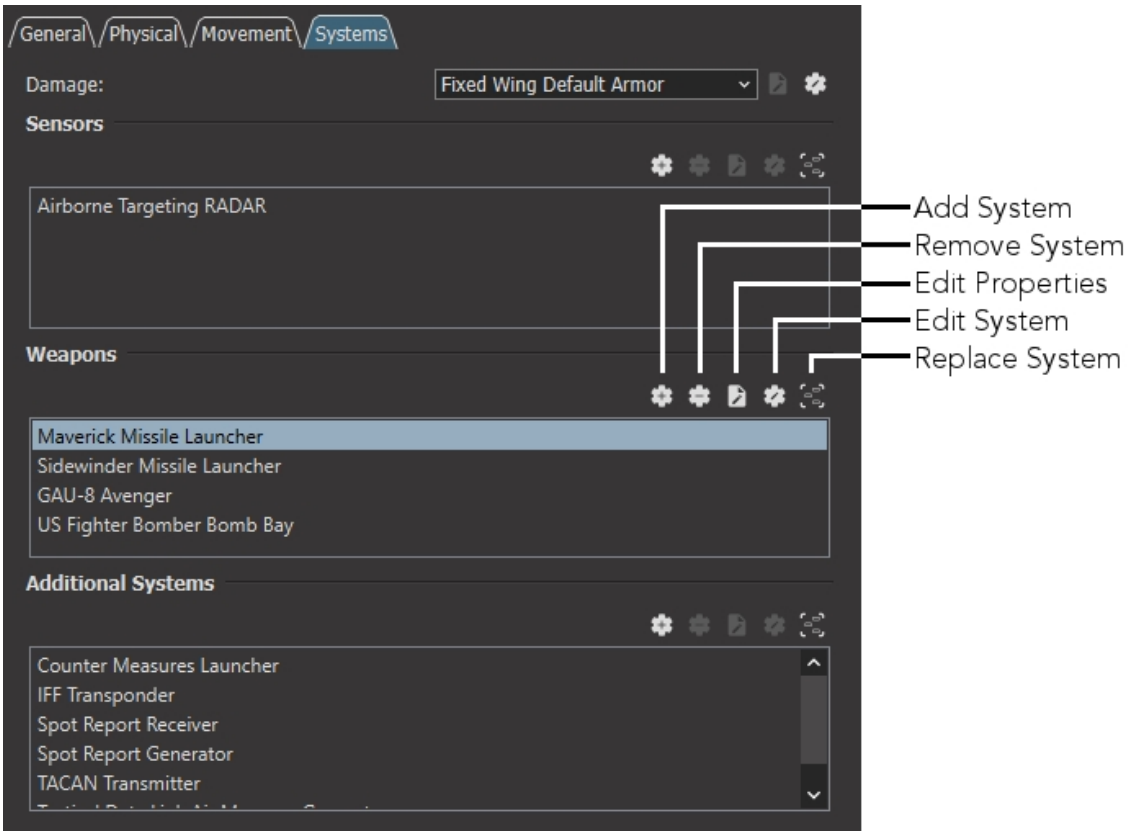
➤ **To remove a custom articulated part:**

1. Select the simulation object.
2. Select the Physical tab.
3. In the Articulated Parts section, select the articulated part you want to remove.
4. Click Remove Selected Part ( ). The part is removed from the entity.

70.4. Editing a Simulation Object's Systems

Simulation objects have complex capabilities, such as the ability to move, to take damage, to sense other simulation objects, and to fire munitions. These capabilities are organized as systems. You can quickly change how a simulation object functions by adding, removing, and changing its systems. The Simulation Object Editor lists the systems available in each category and only lists systems that are appropriate for the entity type (Figure 577). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

Figure 577. Systems tab



Some systems are interdependent. For example, if you add a weapon system to a simulation object, it probably cannot use it unless it also has a sensor system so that it can detect simulation objects. If you change or add systems, it is up to you to know if they have dependencies.

The components that make up a system have parameters and you can edit the parameters of system components by editing the properties of the system.

Use the Simulation Object Editor to edit these types of systems:

- Damage. You can specify the type of damage model that a simulation object uses. When you select a damage model, the Simulation Object Editor configures the damage files that are appropriate for the object type.

- Sensors. You can specify the sensors that a simulation object uses to locate other simulation objects in the simulation. (By contrast, sensor signatures, which you can also set, specify the distance at which other entity sensors can locate this simulation object. For a list of the sensor signatures used for VR-Forces by category, see Table 73 on page 1508.) When you add a weapon, the Simulation Object Editor configures the components and connections necessary for the sensor to work.
- Weapons. The Simulation Object Editor lists the weapon systems that are appropriate for the object type. When you add a weapon, the Simulation Object Editor configures the components, targets, ammunition, and connections necessary for the weapon to work.
- Movement. You can specify the type of movement system that a simulation object uses, for example, tracked, wheeled, pedestrian, and so on. When you select a movement system, the Simulation Object Editor configures all of the actuators and controllers that the simulation object needs to implement the system.
- Additional Systems. This category includes spot reports, IFF transponders, emergency transponders, and refueling systems.

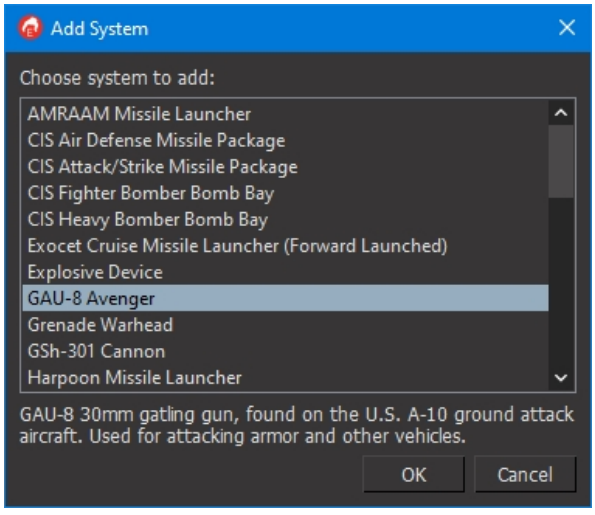
70.4.1 Adding a System

There is no specific limit to the number of sensors, weapons, or other systems that you can configure for a simulation object.

➤ **To add a system:**

1. Select a simulation object in the list.
2. Select the Systems tab (Figure 577 on the previous page).
3. Click the Add System button (⚙) for the system type you want to add. The Add System dialog box opens (Figure 578). It lists the systems of this type that are available for the simulation object you are editing. If you select a system in the window, the dialog box displays a brief description.

Figure 578. Add System dialog box



4. Select the system you want to add.
5. Click OK. The system is added to the list in the Simulation Object Editor.

70.4.2 Selecting a Damage Model

You can only have one damage model specified for a simulation object.

➤ To specify a damage model, on the Systems tab (Figure 577 on page 1390), select an option from the Damage list.

70.4.3 Selecting a Movement Model

You can only have one movement model specified for a simulation object.

➤ To specify a movement model, on the Movement tab (Figure 571 on page 1381), select an option from the Movement Dynamics list.

70.4.4 Editing a System's Properties

Some systems have properties that you can edit.

➤ **To edit a system's properties:**

1. Select the simulation object whose system you want to edit.
2. Select the Systems tab (Figure 577 on page 1390).
3. For sensors, weapons, or other systems, select the system whose properties you want to edit. (You do not have to select the damage or movement model, because there can only be one.) If the system has properties, the Properties button is enabled.
4. Click the Edit Properties button ( ). A Properties dialog box opens. The 3D model in the Visual Model Panel displays a red dot on the simulation object where the system is located. In some cases there are multiple locations.
5. Edit the properties as desired.
6. Click OK.

70.4.5 Removing a System

You can remove sensor, weapon, and additional systems.

⚠ If you remove a system that another system depends on, the other system will not work. It is your responsibility to know these dependencies.

➤ **To remove a system:**

1. On the Systems tab, select the system you want to remove.
2. Click the Remove System button ( ).

70.4.6 Replacing a System

You can swap systems on a simulation object by selecting a system you do not want and replacing it with one that you do. This writes the new system entry with the same string-id that the existing one had and simply hides the existing one. For a simulated object that you created as the child of another (described in "69.3.2 Inheriting a New Model from an Existing Model" on page 1361), this is a system inherited from the parent.

➤ **To replace a system:**

1. Select a simulation object in the list.
2. Select the Systems tab (Figure 577 on page 1390).
3. Select the system you want to replace.
4. Click the Replace System button ( ). The Add System dialog box opens (Figure 578 on page 1391).
5. Select the system you want the simulation object to have.
6. Click OK. That system is added to the list and the replaced system is removed.

70.5. Creating and Editing Munitions

Munitions fall into two broad categories: simulated munitions and non-simulated munitions. As the name implies, simulated munitions are simulated as individual objects. This category includes missiles, bombs, mines, and hand grenades. Since they are simulated, they have visual and 2D models. They have all of the various tabs for configuring their parameters that other simulation objects have, plus a Firing tab for adjusting their accuracy and a Detonation tab for configuring the damage power.

Non-simulated munitions are not simulated as individual objects. This category is for ammunition that is fired by some of the direct fire weapons, such as rifles, machine guns, and some ballistic rounds. These munitions have a General tab for basic identifying information and a Detonation tab for configuring their damage power.

70.5.1 Creating a Simulated Munition

Simulated munitions have visual models and are configured just like other simulation objects.

➤ **To create a simulated munition:**

1. Choose **Model > New Munition**. The New Munition dialog box opens.
2. Select the New Munition option.
3. Click OK. The Create New Model dialog box opens.
4. In the Platform Type list, select the platform type for the munition. The Description field provides information about the type of munition represented by this platform.
5. In the Template field, select the template you want to use. It is possible that there are no templates for the platform type.
6. In the Name box, enter a name for the munition.

7. Optionally, edit the Object Type.
8. In the Take Visuals list, select the model you want to use for this munition.
9. Click OK. A new entry is added to the munitions list.
10. If applicable, on the General tab, specify the object types that this munition can target.
11. Edit the Detonation power, as described in "78.2.2 Configuring Detonation Power in the SOE" on page 1550.
12. Edit other parameters for this munition as desired.

70.5.2 Displaying Munition Detonations

There is a button for simulated munitions that allows you to preview the detonation in different environments.

➤ **To display the detonation for simulated munitions:**

1. Select the Munitions tab.
2. Select the munition.
3. Click the new Detonate button () in the 3D display. The Display Detonations dialog box opens.
4. Select the detonation type you want to display.
5. Click the Detonate button.

70.5.3 Creating a New Ammunition Type

Ammunition munitions are not simulated as individual objects.

➤ **To create a new ammunition type:**

1. Choose **Model > New Munition**. The New Munition dialog box opens.
2. Select the New Ammunition option.
3. Click OK. The Add New Ammunition dialog box opens.
4. Specify the object type for the new ammunition by selecting options from the lists of typing the object type directly.
5. Click OK.
6. Click OK. A new entry is added to the munitions list.
7. On the General tab specify the object types that this munition can target.
8. If this munition has a tracer type, specify it on the General tab.
9. Edit the Detonation power, as described in "78.2.2 Configuring Detonation Power in the SOE" on page 1550.

70.6. Configuring Wakes

You can configure the wake parameters for surface entities in the Simulation Object Editor. The model window shows the ocean state and you can vary the wind speed, entity speed, and choppiness to see how the wake looks under varying ocean conditions (Figure 579).

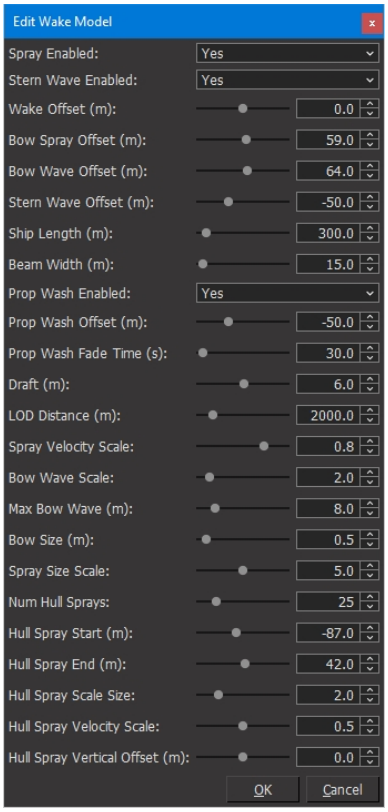
Figure 579. Wake display



➤ **To configure wakes:**

1. Select a surface entity in the object list.
2. Click the Edit Wake button () on the Visual Models Panel (Figure 555 on page 1346). The Edit Wake Model dialog box opens (Figure 580).

Figure 580. Edit Wake Model dialog box



3. Change wake parameter values as desired. For details about wake parameters, see "Configuring Wakes" in the *Adding Content to MAK ONE Applications Reference Manual*.
4. To see the effect of parameter changes under different sea conditions:
    a. Click the Environment Settings button () on the Visual Models Panel. The Environment Settings dialog box opens.
    b. Adjust the wind speed, wind direction, choppiness, and ocean quality values.
    c. Click Close.
5. Click OK.

> To view the wakes and propeller wash in the Visual Models Panel, choose **View > Visual Effects** and then adjust the speed.

70.7. Creating Randomized Simulation Objects

A randomized simulation object is an object that you can use to quickly add a generic simulation object type, such as a civilian vehicle or commercial aircraft, using randomly chosen specific models of these object types. For example, if you want to add several cars to a scenario, you could individually select a Chevrolet Caprice, a Ford Escape, a Toyota

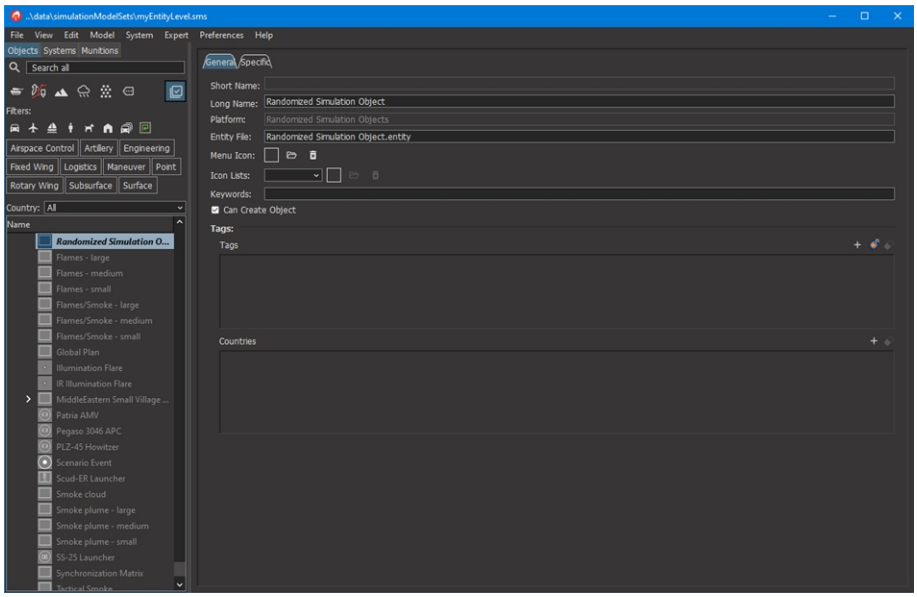
Pickup, and so on, and add them to the scenario. Or you could select a randomized simulation object that references a list of civilian vehicles and click multiple times on the terrain. With each click, a randomly chosen vehicle is added to the scenario.

You create randomized simulation objects in the Simulation Object Editor. You can create them in entity-level and aggregate-level scenarios.

➤ **To create a randomized simulation object:**

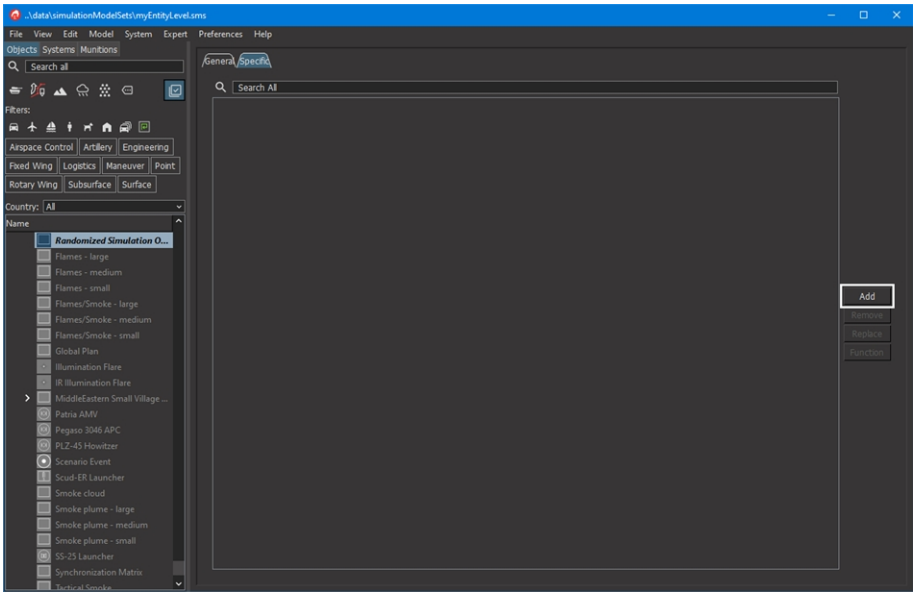
1. Start the Simulation Object Editor.
2. Load the SMS you want to use.
3. Choose **Model > New Randomized Simulation Object**. The Randomized Simulation Object dialog box opens.
4. Enter a name for the simulation object and then click OK. The new object is added to the object list.
5. Select the new object in the object list (Figure 581).

Figure 581. Randomized simulation object



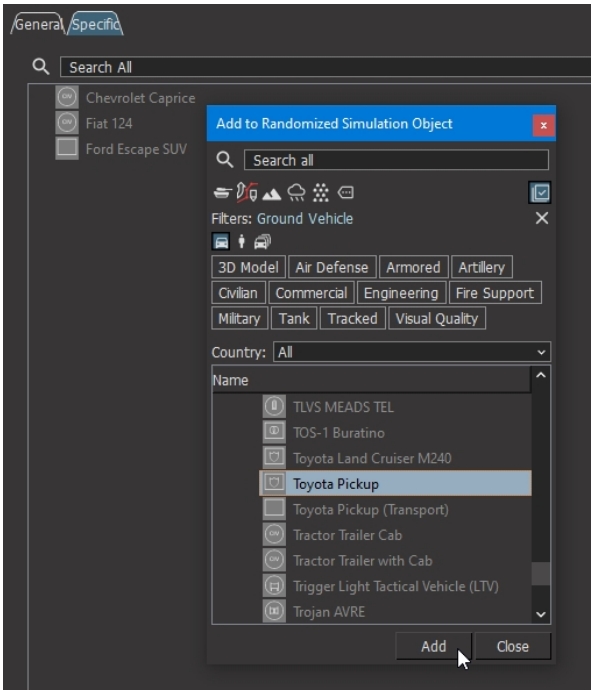
6. Optionally, edit the common simulation object attributes, such as tag, palette, and so on.
7. Select the Specific tab (Figure 582).

Figure 582. Randomized object, Specific tab



8. Click the Add button (Figure 582). The Add to Randomized Simulation Object dialog box opens.
9. Select the simulation objects that you want to be part of this randomized simulation object. To select an object, double-click it or select it and click Add. As you select objects, they are added to the list (Figure 583). If you want the probability of a particular simulation object being created to be higher than others, add multiple instances of the same simulation object to the list.

Figure 583. Randomized object list



10. Click Close.
11. Save the SMS.

70.8. Configuring Scripted Entity Movement

In most cases, simulation objects move based on the intelligence built into their components – sensors, controllers, and actuators. However, there are two instances in which entity movement follows a predetermined sequence, or script. These cases are ballistic missile movement and animated movement tasks.

When you launch a ballistic missile (by creating a missile target in VR-Forces), it does not follow a route or do any path planning; it flies based on the parameters in the scripts it is configured to use. Similarly, if you give an animated movement task to an entity, it moves in whatever way the script prescribes.

Ballistic missile movement is configured using comma-delimited value (CSV) files. Animated movement tasks can be configured with CSV files or 3DS ASCII Export (ASE) files. We refer to these files generically as movement files.

A movement file specifies a time sequence and a set of parameters that define the entity's movement for each time entry. Ballistic missile movement files define lateral-distance, altitude, and range. Animated movement movement files define lateral distance, altitude, range, heading, pitch, and roll.

> Ballistic missile movement files are configured in a dialog box that is shared with animated movement movement files ("70.8.2 Configuring Animated Movement" on page 1401). Animated movement movement files have more parameters than ballistic missile movement files. If a ballistic missile is configured with a movement file that has these additional parameters, they are ignored.

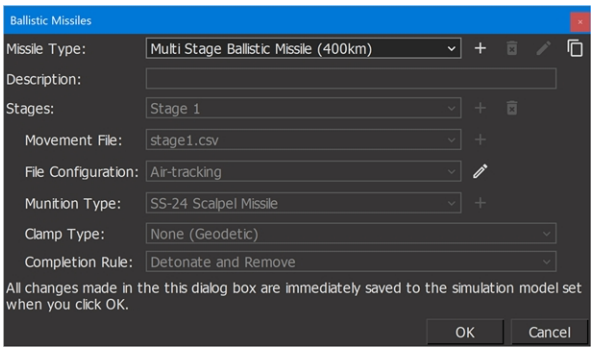
70.8.1 Configuring Ballistic Missile Movement

When you create a Missile Target event in VR-Forces, you specify a missile type. Missile types are configured in the Simulation Object Editor.

➤ **To configure ballistic missiles:**

1. Choose **Edit > Ballistic Missiles**. The Ballistic Missiles dialog box opens (Figure 584).

Figure 584. Ballistic Missiles dialog box



2. In the Missile Type list, select the missile type that you want to edit. You can add, delete, and rename missile types using the Add (+), Delete (🗑), and Rename (✎) buttons.
3. Optionally, in the Description text box, enter a description. This description is used as a tooltip for the Missile Type in the Create Missile Target and Edit Missile Target dialog boxes.
4. In the Stages list, select the stage that you want to configure. Missiles can be single stage or multi-stage. For single stage missiles, configure Stage 1. For multi-stage missiles, use the Add and Delete buttons to add or delete stages.

   For each stage, specify:

   — Movement File. The CSV file to use. The list lists the files in *./data/simulationModelSets/<model_set>/scriptedObjectMovement*.

   If you want to use a CSV file that is not listed, click the Add button (+) next to the Movement File list. A file chooser dialog box opens, where you can select the CSV file that you want to use. The file is copied to *./data/simulationModelSets/<model_set>/scriptedObjectMovement*.

   — File Configuration. Select a file configuration from the list. File configurations specify the type of value in each column in the movement file. For details, see "70.8.3 Editing Movement File Configurations" on page 1403.

   — Munition Type. The missile to use.

   — Clamp Type. Determines how to calculate altitude for the animation. Table 57 describes the clamp type options.

   — Completion Rule. The action to take when the movement in the CSV file is complete.
5. Click OK.

Table 57. Clamp type options

Option	Description
None (Geodetic)	Altitude in the animation is used as altitude above the altitude of the animation reference location. A constant altitude in the animation results in a path that follows the curvature of the earth, when a round earth terrain is being used.
None (Topographic)	Altitude in the animation is used as altitude above a plane defined by the animation reference orientation. A constant altitude in the animation results in a straight path, even as the earth curves, when a round earth terrain is being used.
High Fidelity (Including Orientation)	Altitude in the animation is ignored, and the object is ground clamped to the terrain using three support points, which also sets the orientation of the object.
Low Fidelity (Terrain Clamp Only)	Altitude in the animation is ignored, and the object is ground clamped to the terrain using only its origin point. Orientation is set solely from the animation.
Altitude Above Terrain (High Fidelity)	Altitude in the animation is used as the altitude above the terrain height.
Ground Clamp if Altitude Offset is 0 (High Fidelity)	Acts like **High Fidelity** for frames where altitude is 0, and acts like **None (Geodetic)** for frames where altitude is non-zero.
Radar Face Cartesian	Similar to **None (Topographic)**, except the Y coordinate is inverted. The animation would typically be started with an orientation reference corresponding to the angle of a radar face. Useful for cases where animation data is captured radar data.

70.8.2 Configuring Animated Movement

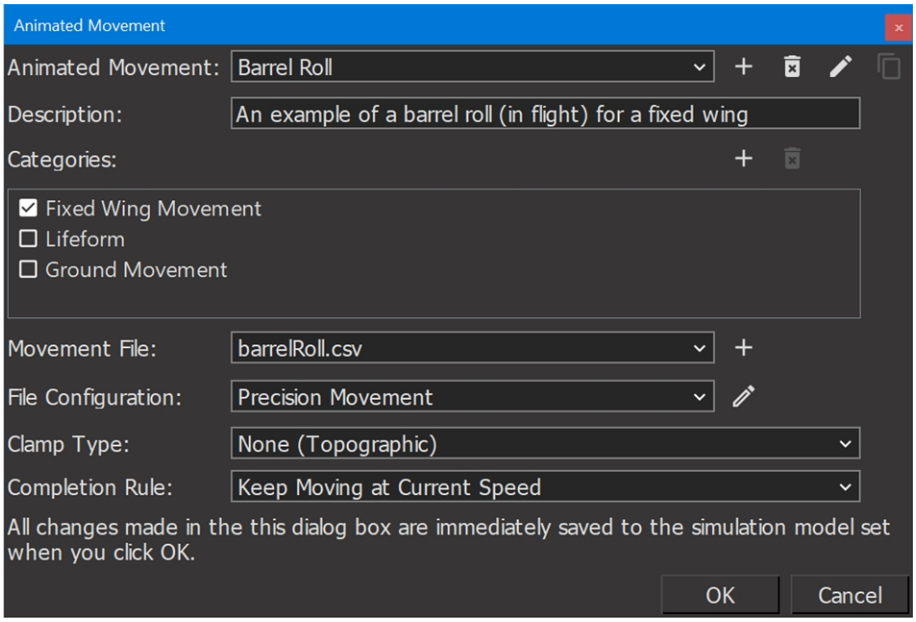
The animated movement feature lets you script a sequence of entity movements. You can do this using a CSV file or a 3DS ASCII Export (ASE) file. Typically, movement are created using telemetry data. VR-Forces includes some sample animated movement files. You configure the animated movement options in the Simulation Object Editor.

Animated movement is assigned to an entity using the Animated Movement task, which causes an entity to follow a predefined path defined in an external file. The entity moves between extrapolated points without the ability to respond to external stimuli. For example, this could be illustrated by a ballistic missile flying along a prerecorded telemetry data path.

➤ **To configure animated movement:**

1. Choose **Edit > Animated Movement**. The Animated Movement dialog box opens (Figure 585).

Figure 585. Animated Movement dialog box



2. To edit an existing animated movement, select it in the Animated Movement list. To create a new animated movement:
    a. Click the Add button (+) next to the Animated Movement list. The New Configuration dialog box opens.
    b. Enter a name for the new animated movement.
    c. Click OK.
3. Optionally, enter a description for the animated movement. This description is displayed in the Animated Movement dialog box when an Animated Movement task is assigned.
4. In the Categories window, select the categories that this animated movement applies to. Use the Add and Delete buttons to add and delete categories.
5. In the Movement File list, select the movement file to use. The list lists the files in *./data/simulationModelSets/<model_set>/scriptedObjectMovement*. If you want to use a CSV file or ASE file that is not listed:
    a. Click the Add button (+) next to the Movement File list. A file chooser dialog box opens.
    b. Select the file that you want to use. The file is copied to *./data/simulationModelSets/<model_set>/scriptedObjectMovement*.
6. If you chose an ASE file, the dialog box displays a field called Animation. Select an animation from the list.

    If you chose a CSV file, it displays a field called File Configuration. Select a file configuration from the list. File configurations specify the type of value in each column in the movement file. For details, see "70.8.3 Editing Movement File Configurations" on the facing page.

7. In the Clamp Type list, select an option for calculating the altitude. For details, see Table 57 on page 1401.
8. In the Completion Rule list, select the action you want the entity to take after it completes the animated movement task.

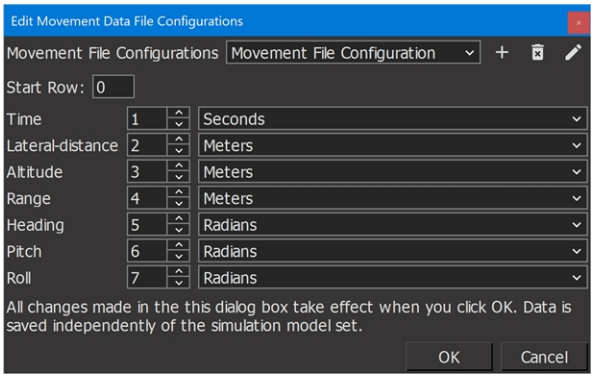
70.8.3 Editing Movement File Configurations

VR-Forces assumes that a ballistic missile CSV file contains four columns of data that describe time, lateral distance, altitude, and range. It assumes that animated movement files have seven columns that describe time, lateral distance, altitude, range, heading, pitch, and roll. VR-Forces includes file configurations for the sample CSV files provided. They are organized in the order that the columns are listed in this paragraph. If a CSV file you are using is organized differently, you must create a file configuration that matches your file or edit one of the provided configurations.

➤ **To create or edit a movement file configuration:**

1. In the Ballistic Missiles dialog box or the Animated Movement dialog box, click the Edit button (✎) next to the File Configuration list. The Edit Movement Data File Configurations dialog box opens (Figure 586).

Figure 586. Edit Movement Data File Configurations dialog box



- Ballistic missile file configurations have four rows. Animated movement file configurations have seven rows. If you specify a seven-row configuration for a ballistic missile, it ignores the values that it does not use.
- If you are editing an animated movement that uses an ASE file, there is no File Configuration option. This only applies to CSV files.

2. In the Movement File Configurations list, select the configuration you want to edit. If you are creating a new configuration, click the Add button (+). The New Configuration dialog box opens.
   a. Enter a name for the configuration.
   b. Click OK.

3. Optionally, change the Start Row value. This is the row in the CSV file from which VR-Forces should start reading values for entity movement.
4. Specify the column in the CSV file that has the Time value.
5. Optionally, select the time unit from the list.
6. Repeat steps 4 and 5 for each movement parameter.
7. Click OK.



71. Editing Units for Entity-Level Scenarios

This chapter explains how to edit units for entity-level scenarios (*EntityLevel.sms*) in the Simulation Object Editor.

71.1. Introduction to Editing Units 1406
71.2. Creating a Unit for Entity-Level Scenarios 1406
71.3. Specifying an Aggregate As Option 1406
71.4. Editing Unit Composition 1407
- 71.4.1 Adding Subordinates 1408
- 71.4.2 Removing Subordinates 1409
- 71.4.3 Replacing Subordinates 1409
- 71.4.4 Changing the Order of Subordinates 1410
- 71.4.5 Editing a Subordinate Unit 1410
- 71.4.6 Defining Functions for Subordinates 1410

71.5. Editing Formations 1411
- 71.5.1 Adding Formations 1412
- 71.5.2 Removing Formations 1413
- 71.5.3 Renaming Formations 1413
- 71.5.4 Specifying the Default Formation 1413
- 71.5.5 Expanding and Collapsing the Display of Formations 1413
- 71.5.6 Displaying Bounding Boxes 1414
- 71.5.7 Copying Formations 1414
- 71.5.8 Laying Out a Formation Automatically 1415
- 71.5.9 Changing a Formation's Layout by Hand 1415

71.1. Introduction to Editing Units

VR-Forces can simulate units in two different ways:

- In entity-level scenarios, VR-Forces models the individual entities of a unit. Units for entity-level scenarios are configured in *EntityLevel.sms* or SMSs based on it.
- Units in aggregate-level scenarios do not model individual entities. Combat operations are data-driven and are not based on individual entity interactions. Units for aggregate-level scenarios are configured in *AggregateLevelBase.sms*, *AggregateLevel.sms*, or SMSs based on them.

Units in entity-level scenarios and aggregate-level scenarios have many of the same parameters as individual entities. Therefore, for the most part, configuring them in the Simulation Object Editor involves the same procedures as those described in "Chapter 68. The Simulation Object Editor and SMSs" on page 1303 and "Chapter 70. Editing Entities for Entity-Level Scenarios" on page 1375.

The two major differences in configuring the different types of units is that units in entity-level scenarios have subordinates and formations and do not have sensors. Units in aggregate-level scenarios do not move in formation. They have sensors and they have a large amount of information about their combat capabilities, personnel, equipment, and supplies.

71.2. Creating a Unit for Entity-Level Scenarios

➤ **To create a unit for entity-level scenarios:**

1. Follow the procedures for creating a new entity in "69.3. Creating a New Simulation Object" on page 1360.
2. Add subordinates, as described in "71.4.1 Adding Subordinates" on page 1408.
3. Optionally, configure formations.

71.3. Specifying an Aggregate As Option

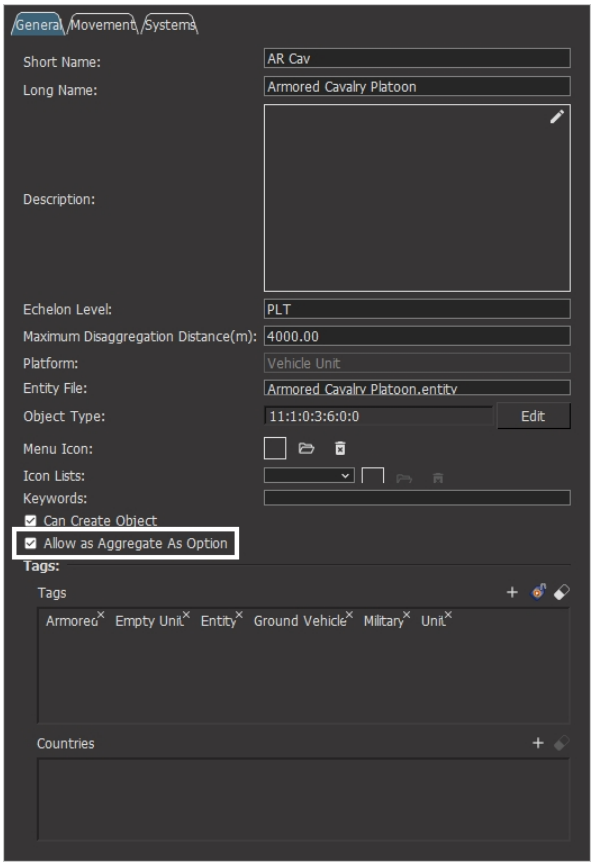
When you create a unit using the Aggregate As command, the Aggregate As dialog box has a list of unit types that you can select for the unit that is to be created. VR-Forces is configured with just a short list of generic unit types. You can specify that any unit configured in the Simulation Object Editor be included in the list of Aggregate As options.


This procedure is valid for entity-level and aggregate-level SMSs.

➤ **To specify that a unit can be an option in the Aggregate As list:**

1. Select the unit in the object list.
2. Select the General tab (Figure 587).

Figure 587. General tab, entity-level unit



3. Select the Allow as Aggregate As Option check box.
4. Save the SMS.

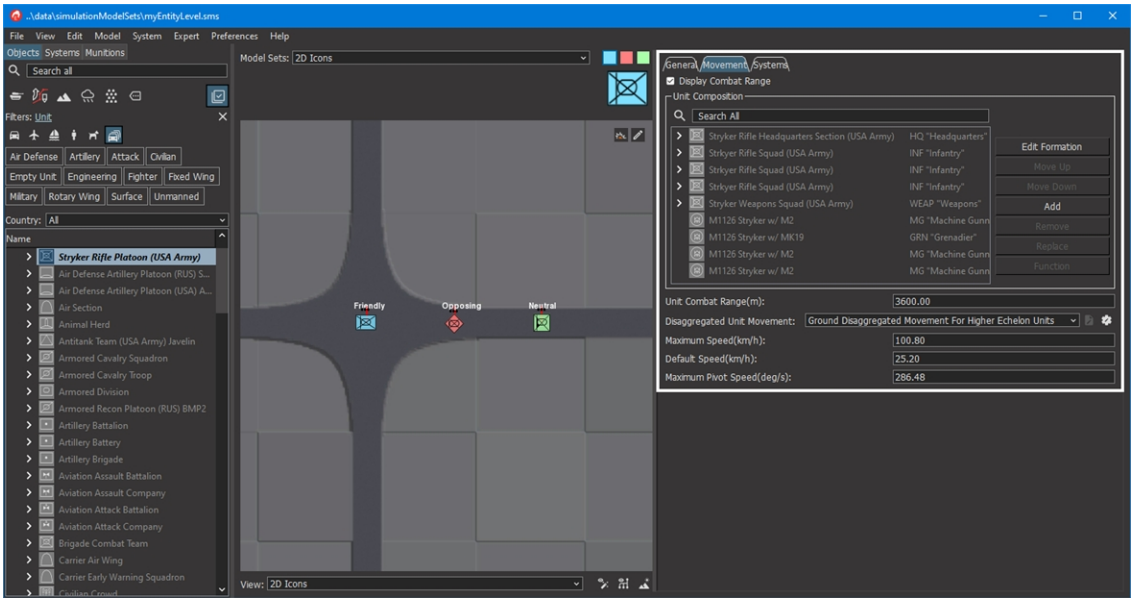
71.4. Editing Unit Composition

Many units available in VR-Forces have predefined sets of subordinates, which you can modify. (Some units are configured as "empty" units — they have no subordinates. These empty units are still available to the list of choices in the Aggregate As dialog box and for mapping to units when you import MSDL scenarios.)

> A few units (and groups that are managed like units) do not allow aggregation — that is, you cannot combine the subordinates into a single object. These include Crowds, Convoys, Carrier Air Wings, and Animal Herds.

Figure 588 illustrates the Simulation Object Editor with a unit selected. The Unit Composition group box is highlighted.

Figure 588. Unit Composition



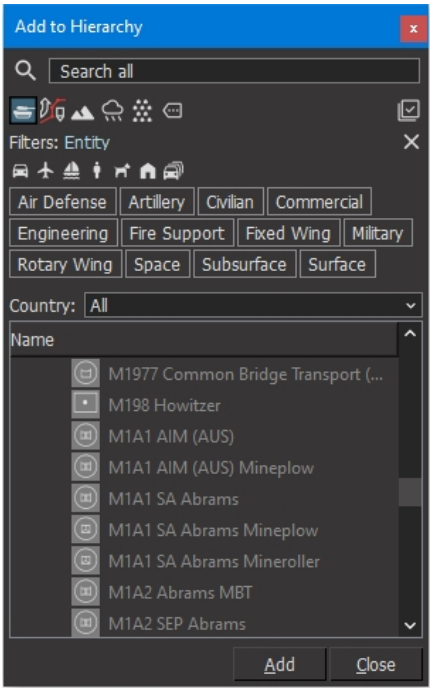
71.4.1 Adding Subordinates

When you create a new unit without copying an existing unit, you must specify its subordinates. You can also edit subordinates for existing units.

➤ **To add a subordinate to a unit:**

1. In the Objects List, select the unit.
2. Select the Movement tab.
3. In the Unit Composition group box (Figure 588 above), click Add. The Add to Hierarchy dialog box opens (Figure 589).

Figure 589. Add to Hierarchy dialog box



4. Optionally, click the Entity filter list.
5. In the list of simulation objects, select the simulation object that you want to add.
6. Click Add. The simulation object is added to the list in the Unit Composition window. It is also added to each formation using a default layout algorithm.
7. Continue to add as many subordinates as desired.
8. Click Close.

71.4.2 Removing Subordinates

➤ **To remove a subordinate from a unit:**

1. In the Unit Composition window (Figure 588 on the previous page), select the subordinate that you want to remove.
2. In the Unit Composition group box, click Remove. The subordinate is removed from the unit.

71.4.3 Replacing Subordinates

Replacing a subordinate is a convenient way to remove a subordinate and insert a different subordinate in the same location in one step.

➤ **To replace a subordinate:**

1. In the Unit Composition window (Figure 588 on the previous page), select the subordinate that you want to replace.

2. In the Unit Composition group box, click Replace. The Add to Hierarchy dialog box opens (similar to Figure 589 on the previous page).
3. In the list of simulation objects, select the object that you want to add as a replacement.
4. Click OK. The new object replaces the selected simulation object in the Unit Composition window. It is also added to each formation in the location of the replaced simulation object.

71.4.4 Changing the Order of Subordinates

The order of subordinates in the Unit Composition window determines which entity is the leader, the order in which units are placed into formation, and the succession when units are reorganized.

➤ **To change a subordinate's position in the unit composition:**

1. In the Unit Composition window (Figure 588 on page 1408), select the subordinate that you want to move.
2. In the Unit Composition group box, click Move Up or Move Down.

71.4.5 Editing a Subordinate Unit

Units can contain entities and other units as subordinates. When you are editing a unit, you might decide that you want to edit one of its subordinates. You can jump to the subordinate quickly without selecting it in the main list of simulation objects.

> ⚠ If you edit a subordinate unit, the changes affect all uses of that subordinate, not just the one in this particular unit.

➤ **To edit a subordinate:**

1. In the Unit Composition window or in the Formation Editor, right-click the subordinate that you want to edit. A menu displays.
2. In the menu, choose **Edit**. The Simulation Object Editor refreshes to show the selected subordinate. (If the Formation Editor is open, you must close it before you can edit the subordinate.)

71.4.6 Defining Functions for Subordinates

You can specify a subordinate's function within the unit. These are text-based handles that identify a role or function that the subordinate performs in that unit, which are then used in unit behaviors. Some behavior tasks are included with VR-Forces, which are described in "Chapter 33. Unit Behavior Tasks for Entity-Level Scenarios" on page 667.

➤ **To assign a function to a subordinate:**

1. In the Unit Composition window, select the subordinate.
2. Click Function. The Assign Subordinate Function dialog box opens.

3. Select the function you want to assign to the subordinate.
4. Click OK.

Advanced developers can use the VR-Forces behavior engine to write their own behavior trees and make use of functions; see "38.1.2 Unit-Specific Extensions to Standard Behavior Trees" on page 817 for details. You can define a function for a behavior you are scripting if it does not fit into one of the available functions.

➤ **To define a function:**

1. Choose **Edit > Subordinate Functions**.
2. Click the Add button. A row is added to the bottom of the list.
3. Select the Handle field. Enter an abbreviation for the function.
4. Select the Display Text field. Enter the display text for the function.
5. The Model Set specifies the location of the subordinate functions for the SMS you are working in. You cannot edit this field.
6. Click OK.

71.5. Editing Formations

Each unit in *EntityLevel.sms* has the following defined formations:

- Column
- Line
- Vee
- Wedge

> The Convoy unit provided with VR-Forces is an exception to the discussion of units in this section. It does not have any formations because there is no preconceived membership in a convoy.

You can add, remove, and rename formations. You can also:

- Change the layout of subordinates in a formation.
- Create new formations.
- Specify the default formation.

The Formation Editor allows you to change the display of formations:

- Expand and collapse the display of subordinate units.
- Zoom in and out on the display.
- Display subordinate bounding boxes.

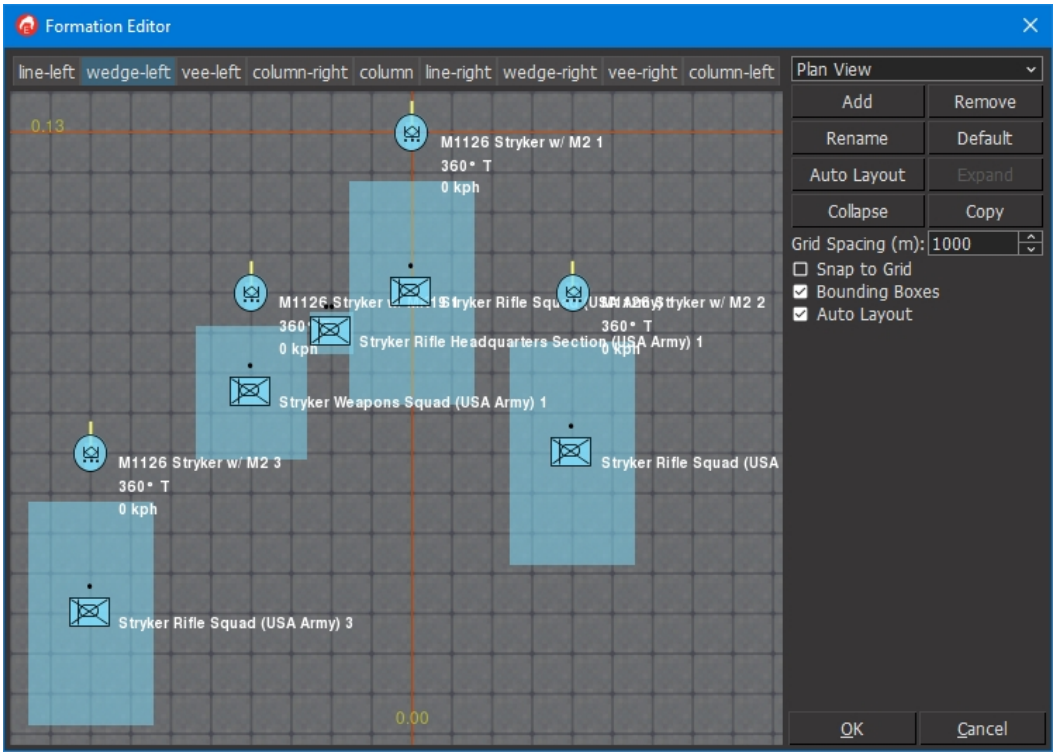
You can also edit formations in formation files. For details, see "79.1. Configuring Formations" on page 1566.

71.5.1 Adding Formations

➤ **To add a formation:**

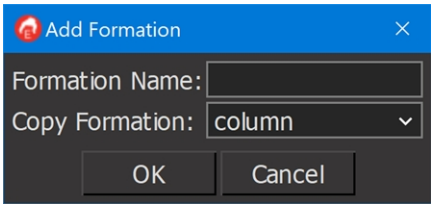
1. In the Unit Composition group box (Figure 588 on page 1408), click Edit Formation. The Formation Editor opens (Figure 590).

Figure 590. Formation Editor



2. In the Formation Editor, click Add. The Add Formation dialog box opens (Figure 591).

Figure 591. Add Formation dialog box



3. In the Formation Name box, enter a name for the formation.
4. In the Copy Formation list, select an existing formation to copy the new formation from.
5. Click OK.
6. Edit the layout of the formation as described in "71.5.9 Changing a Formation's Layout by Hand" on page 1415.

71.5.2 Removing Formations

➤ **To remove a formation:**

1. In the Unit Composition group box (Figure 588 on page 1408), click Edit Formation. The Formation Editor opens (Figure 590 on the previous page).
2. In the Formation Editor, select the tab for the formation that you want to remove.
3. Click Remove. The tab is removed.

71.5.3 Renaming Formations

➤ **To rename a formation:**

1. In the Unit Composition group box (Figure 588 on page 1408), click Edit Formation. The Formation Editor opens (Figure 590 on the previous page).
2. In the Formation Editor, select the tab for the formation that you want to rename.
3. Click Rename. The New Name dialog box opens.
4. Enter a name for the formation.
5. Click OK.

71.5.4 Specifying the Default Formation

The default formation is the formation assigned to a unit when you create it in VR-Forces. The default formation is the leftmost tab in the Formation Editor.

➤ **To specify the default formation:**

1. In the Unit Composition group box (Figure 588 on page 1408), click Edit Formation. The Formation Editor opens (Figure 590 on the previous page).
2. In the Formation Editor, select the tab of the formation that you want to specify as the default formation.
3. Click Default. The selected tab moves to the leftmost position.

71.5.5 Expanding and Collapsing the Display of Formations

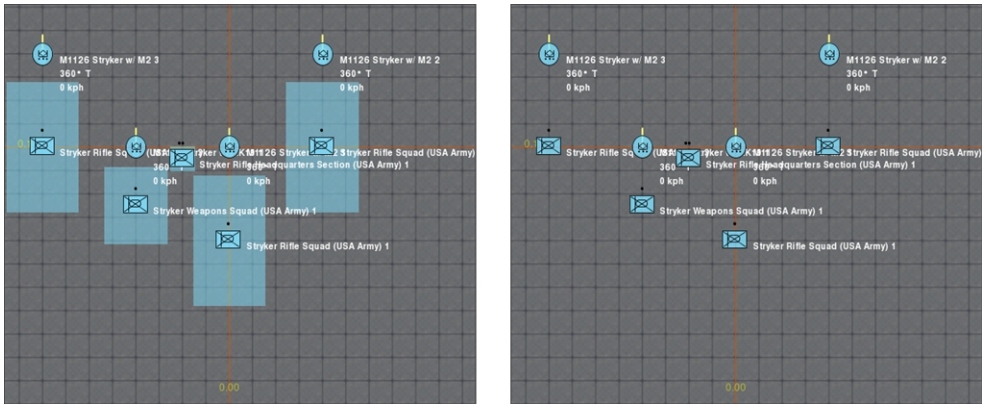
You can expand and collapse the display of formations in the Formation Editor window. This enables you to see how simulation objects are dispersed over the terrain. It can help you quickly identify any unintended variations in subordinate formations.

➤ To expand or collapse the display of formations in the Formation Editor, click the Expand or Collapse buttons.

71.5.6 Displaying Bounding Boxes

You can display bounding boxes for the subordinate units in a unit. This may be useful when you collapse the views of individual units, but you still want to have a sense for the area that they occupy. Figure 592 illustrates a unit with the display of bounding boxes off and on.

Figure 592. Unit bounding boxes in Formation Editor - off and on



➤ To display or hide bounding boxes, in the Formation Editor (Figure 590 on page 1412), select or clear the Bounding Boxes check box.

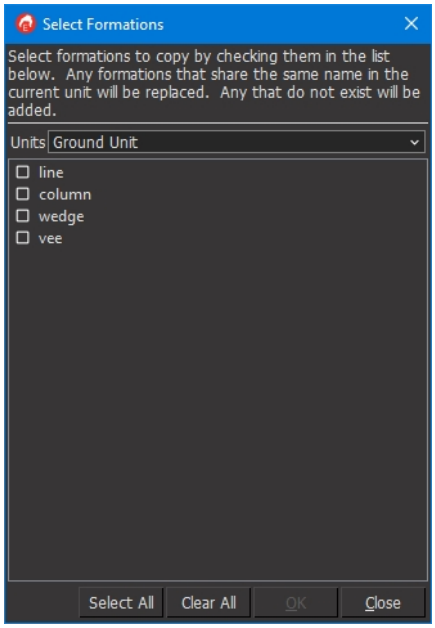
71.5.7 Copying Formations

You can copy formations from one unit to another. When you copy formations, the subordinates in the unit are laid out in each formation according to their order in the Unit Composition window.

➤ **To copy a formation:**

1. In the object list, select the unit that you want to copy formations to.
2. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 590 on page 1412).
3. In the Formation Editor, click Copy. The Select Formations dialog box opens (Figure 593).

Figure 593. Select Formations dialog box



4. In the Units list, select the unit whose formations you want to copy.
5. In the window, select the formations that you want to copy.
6. Click OK. The formations are copied from the selected unit to the currently displayed unit.

71.5.8 Laying Out a Formation Automatically

You can automatically distribute the simulation objects in a unit into one of the built-in formations. You can specify the distance between each entity in the formation.

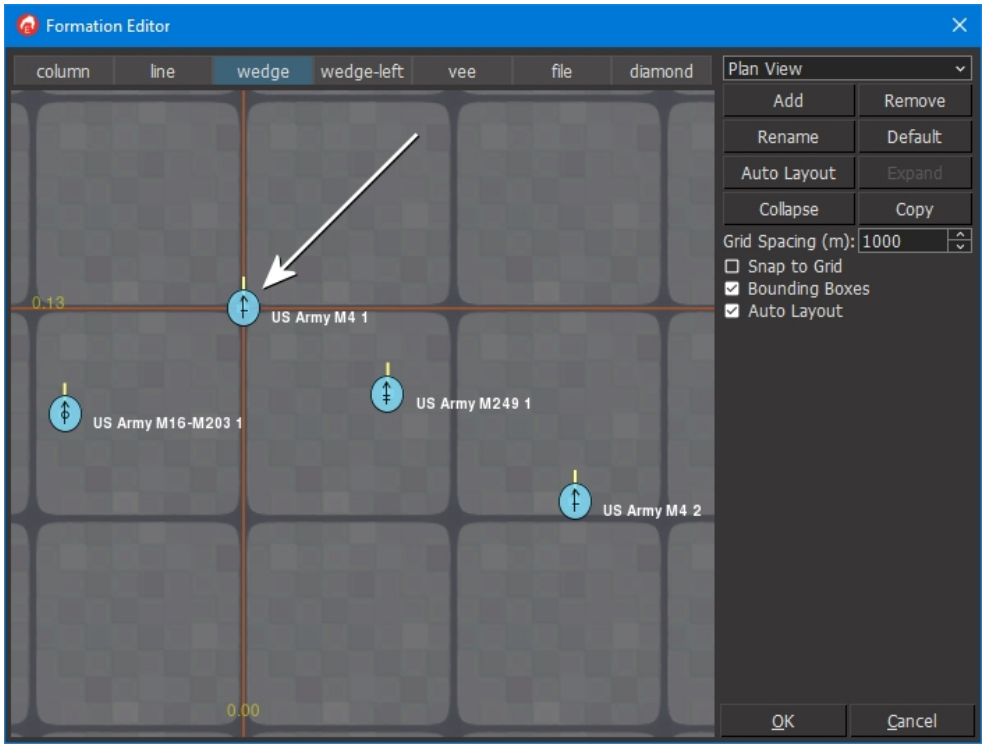
➤ **To lay out a formation automatically:**

1. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 590 on page 1412).
2. In the Formation Editor, select the tab for the formation that you want to lay out.
3. Click Auto Layout. The Auto Layout dialog box opens.
4. In the list, select a formation type.
5. Specify the distance, in meters, between each entity in the formation.
6. Click OK. The simulation objects in the unit are reorganized to the new layout.

71.5.9 Changing a Formation's Layout by Hand

You can lay out a formation by hand. The gridline in the Formation Editor indicates the center of the formation (Figure 594). This is where the unit icon is placed when the unit is collapsed.

Figure 594. Formation Editor cross-hairs



➤ **To change a formation's layout:**

1. In the Unit Composition group box, click Edit Formation. The Formation Editor opens (Figure 590 on page 1412).
2. In the Formation Editor, select the tab for the formation that you want to lay out.
3. To move a subordinate, double-click it and then move it to the location where you want it to be.
4. Click again to place it in the new location.

Snapping Subordinates to the Grid

When you lay out a formation, you can specify that simulation objects snap to the grid. That is, they can only be placed at the intersection of vertical and horizontal grid lines. When snap-to-grid is enabled, if you want to place simulation objects more closely together than the grid allows, you can change the scale factor.

➤ To enable snap-to-grid for entity placement, in the Formation Editor, select Snap To Grid.

Changing the Grid Spacing

Grid spacing determines the distance, in meters, between grid lines. The default is 1000 meters.

➤ To change the grid spacing, enter a value in the Grid Spacing box.



72. Editing Aggregate-Level Objects

This chapter explains how to edit simulation objects and combat engineering objects for aggregate-level modeling.

72.1. Creating Objects for Aggregate-Level Modeling 1418
72.2. Configuring Simulation Objects for Aggregate-Level Modeling 1418
72.2.1 Aggregating Units into Higher Echelon Units 1419
72.3. Configuring Ammunition, Weapons, and Equipment 1420
72.3.1 Defining Ammunition, Weapons, and Equipment Items 1422
72.4. Assemblies 1424
72.4.1 Creating Assemblies 1424
72.4.2 Copying an Assembly to Your SMS 1425
72.4.3 Editing Assemblies 1425
72.4.4 Creating an Assembly from a Unit 1426
72.4.5 Adding Assemblies to a Unit 1427
72.4.6 Removing Assemblies 1428
72.5. Rolling Up Assemblies 1428
72.5.1 Creating Roll Up Rules 1428
72.6. Adding Activities 1429
72.7. Object Parameters for Aggregate-Level Modeling 1430
72.7.1 General Parameters 1431
72.7.2 Physical Parameters 1431
72.7.3 Movement Parameters 1432
72.7.4 Sensor Parameters 1434
72.7.5 Attack Parameters 1435
72.7.6 Defense Parameters 1436
72.7.7 Combat Parameters 1437
72.7.8 Electronic Warfare (EW) Parameters 1437
72.7.9 Personnel and Equipment Parameters 1438
72.7.10 Supplies Parameters 1438

72.8. Configuring Combat Engineering Objects 1439
72.9. Editing Reader/Writer Key/Value Pairs 1440

72.1. Creating Objects for Aggregate-Level Modeling

You create new aggregate-level simulation objects and combat engineering objects the same way you create entity-level objects, by creating a new model or creating an object from an existing model.

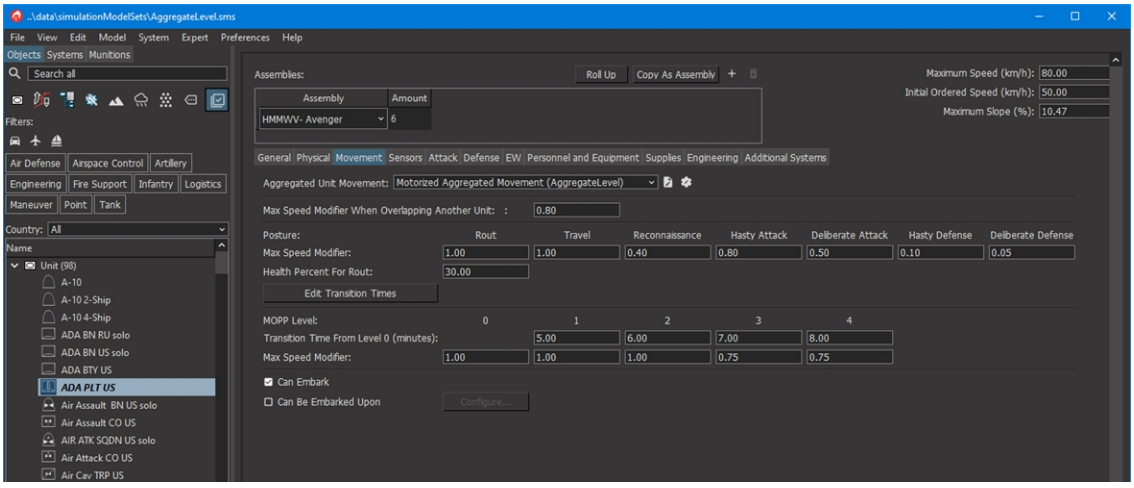
Simulation objects in an aggregate-level scenario have a more complex set of parameters than simulation objects in an entity-level scenario. Ensuring consistency among similar types of simulation objects could become difficult if, after creating new models, you had to configure all of the parameters from scratch, or if, after creating a model from an existing model, you had to back out all of the assemblies and parameters that you wanted to change. Therefore, VR-Forces includes templates for common entity types. When you create a new model from a template (by selecting a template and then choosing **Model > New Model From Existing (Copy)** or **New Model From Existing (Inherit)**), you know that you will have a simulation object with similar characteristics to other simulation objects of its type, and that is ready to customize with assemblies or direct specification of equipment, weapons, and personnel.

See "69.3. Creating a New Simulation Object" on page 1360 to learn more about creating new simulation objects.

72.2. Configuring Simulation Objects for Aggregate-Level Modeling

Simulation objects for aggregate-level modeling (those used in the aggregate warfare model) are configured in the Simulation Object Editor in aggregate-level SMSs like *AggregateLevelBase.sms* and *AggregateLevel.sms*. Some aspects of their configuration, such as name, enumeration, and tags, are the same as for simulation objects in an entity-level scenario. Other parameters, the focus of this chapter, are used in the aggregate warfare model (Figure 595).

Figure 595. Unit in *AggregateLevel.sms*



The parameters for simulation objects in aggregate-level SMSs are organized into these groups:

- General. Basic identification information for the unit.
- Physical. Overall health of the simulation object, its footprint, and sector size for each posture.
- Movement. The movement system to use and how speed is modified by different postures and MOPP levels. (Does not apply to air units.)
- Sensors. Sensor systems and signatures (similar to simulation objects in an entity-level scenario) and how signatures are modified by different postures and MOPP levels.
- Attack. Specifies combat power, ammunition use, weapon systems, and modifiers to combat power. (Does not apply to air units.)
- Defense. Specifies vulnerability to different types of attack and modifiers to vulnerability. (Does not apply to air units.)
- Combat. For air units only. Specifies weapon systems, combat systems, loadout, and other attributes.
- EW. Specifies electronic warfare systems, EW defense, and dependence.
- Personnel and Equipment. Lists personnel and equipment. This information is not used by the sim engine in simulations. It is reported as part of the simulation object's state information.
- Supplies. Specifies the supplies a simulation object needs to move, support personnel, and carry out combat engineering operations.
- Engineering. Lets you specify engineering systems. (Does not apply to air units.)
- Additional Systems. Similar to simulation objects in an entity-level scenario, this tab lets you add other systems such as spot report systems.

The individual parameters in each group are described in "72.7. Object Parameters for Aggregate-Level Modeling" on page 1430.

To configure a simulation object, you specify values for as many of the parameters as are appropriate for the object type. However, you cannot configure ammunition, weapons, and equipment (aircraft, ships, and vehicles), without first defining them.

72.2.1 Aggregating Units into Higher Echelon Units

You can aggregate simulation objects into higher echelon units. And just as with simulation objects in entity-level scenarios, you specify that a simulation object can be listed in the Aggregate As list by selecting the Allow as Aggregate As Option check box. However, the aggregate-level simulation objects that are configured to use the aggregate warfare model do not have the configuration options for specifying subordinates. Therefore, *AggregateLevelBase.sms* has a platform, Aggregate Container, for use when aggregating simulation objects. The SMS has several simulation objects configured with that platform for use when aggregating simulation objects. If you want to create additional hierarchical units for aggregation, it is recommended that you use the Aggregate Container platform.

72.3. Configuring Ammunition, Weapons, and Equipment

To engage in combat, simulation objects need appropriate amounts of ammunition, weapons, and equipment. When you create a simulation object in the Simulation Object Editor, you specify the amounts that it should have by selecting them from lists of the available types of these resources. The items in these lists are also configured in the Simulation Object Editor. You can configure the ammunition, weapons, and equipment for simulation objects directly for each entity, or you can simplify the process by creating assemblies.

For example, a U.S. armored cavalry troop (AR CAV TRP US) has nine M1A2s, thirteen M3A3s, and two M1064 vehicles. An M1A2 has a 120 mm gun, a 12.7 mm machine gun, and two 7.62 mm machine guns. An M3A3 has two TOW missiles, a 25 mm gun, and a 7.62 mm machine gun. An M1064 has a 120 mm mortar and a 12.7 mm machine gun. Therefore, the entity as a whole has twenty-six TOW missiles, thirteen 25 mm guns, nine 120 mm guns, eleven 12.7 mm machine guns, thirty-one 7.62 mm machine guns, and two 120 mm mortars (Figure 596).

Figure 596. Anatomy of a unit without using assemblies



To configure this unit directly, you would go to the Personnel and Equipment tab in the Simulation Object Editor, select all these weapons in the Weapons list, and then enter the amounts for each. You also need to add up all the ammunition required by these weapons and enter them on the Attack tab.

That configuration method works well for simple units and individual entities. If you had to calculate weapons and ammunition at this level of granularity for all units, however, it could become a tedious and error-prone activity. There is an alternative to directly configuring all ammunition, weapons, and equipment individually: you can create

assemblies. Assemblies are resources such as complete weapons and vehicles, or even small units, such as platoons. They are a convenient way to package ammunition, weapons, and equipment to more easily add them to units.

Each assembly configures the ammunition and weapons it needs for one instance of the resource, such as an M1A2. When you create a unit, you simply give it the assemblies it needs. Their capabilities are "rolled up" to define the total amount of resources for the entity.

Figure 597 illustrates the anatomy of an AR CAV TRP US entity that uses assemblies to represent M1A2, M3A3, and M1064 vehicles. Each assembly specifies its weapons and ammunition. When you configure the AR CAV TRP US, you simply specify that it has nine M1A2, thirteen M3A3, and two M1064 assemblies. You roll up the assemblies and the Simulation Object Editor calculates the number of each type of weapon needed (Figure 598).

Figure 597. Anatomy of a unit using assemblies

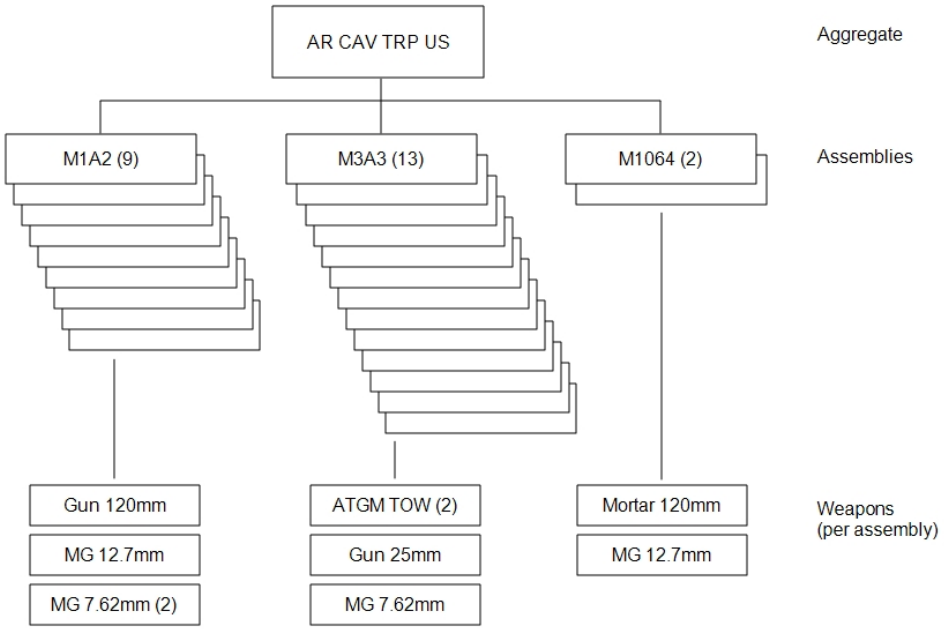
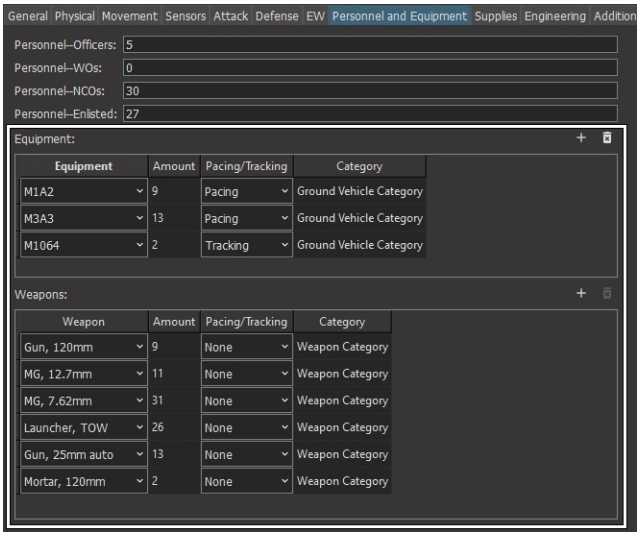


Figure 598. AR CAV TRP US weapons rolled up on Personnel and Equipment tab



Using assemblies, the process for configuring units is to:

1. Define the weapons, equipment, and ammunition that units can use.
2. Define assemblies.
3. Create a unit.
4. Specify the unit's assemblies.
5. Roll up the assemblies.
6. Configure all other parameters.

For more information about assemblies, see "72.4. Assemblies" on page 1424.

72.3.1 Defining Ammunition, Weapons, and Equipment Items

Each ammunition, weapon, and equipment item that you want to be available for use by simulation objects is defined as a key/value pair. The key can be any text string you want to use. There is no relationship to any underlying VR-Forces code. The value is a Display Name – the string used in lists when creating assemblies or adding resources directly to a simulation object.

For each ammunition key/value pair you also must specify a type, where the types are:

- Anti-air
- Anti-personnel
- Anti-ship
- Anti-tank
- Biological
- Chemical
- High explosive

- Nuclear
- For weapon system

Use the anti-air, anti-personnel, anti-tank, and anti-ship categories for ammunition that will be used by assemblies or added directly to a simulation object. Use For Weapon System for any type of ammunition that will be used by a weapon system that must be added to a simulation object, such as a missile system, a guided bomb, or indirect fire system.

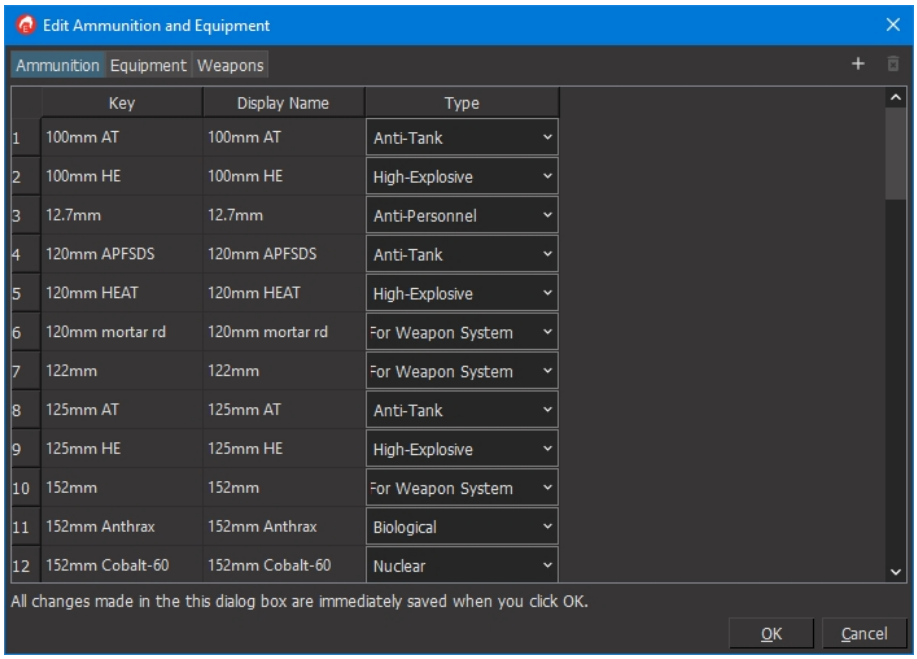
For each weapon and equipment key/value pair, you also must specify a category, where the categories are aircraft, ships, vehicles, and weapons.

The ammunition, weapons, and equipment defined in *AggregateLevelBase.sms* are representative of the weapons and equipment in common use by the United States and Russia. However, you can add any sort of resource you want. For example, you could create entries for knives, slingshots, and shovels. You can add them to assemblies or directly to a simulation object. The only real requirement for making them useful in a scenario is to assign an appropriate level of combat strength when you add them to a simulation object.

➤ **To define an ammunition, equipment, or weapon item:**

1. Choose **Edit > Ammunition and Equipment**. The Edit Ammunition and Equipment dialog box opens (Figure 599).

Figure 599. Edit Ammunition and Equipment dialog box



2. Select the tab for the type of resource you want to add.
3. Click the Add button (+). A row is added to the list. The Key cell is editable.
4. Enter the key value.
5. Click Tab or double-click the Display Name cell to make it editable.

6. Enter the text you want to be visible in the lists where this item will be included.
7. For ammunition items, select an option in the Type list.

   For equipment items, select the appropriate category.

   For weapon items, select Weapon Category.
8. Click OK.

72.4. Assemblies

Assemblies are a convenient way to provide a simulation object with a set of resources without individually configuring each resource.

At the aggregate level, a unit does not model individual entities. It represents some number of vehicles and personnel. For example, a mechanized infantry company has personnel, armored personnel carriers, and so on, which carry a variety of weapons and ammunition. Rather than explicitly configuring a unit to have, for example, 20 APCs, 20,000 rounds of ammunition, 10 howitzers, and 200 infantry, you can configure the ammunition and weapons for one APC as an assembly. Then you would assign a unit 20 APC assemblies. You would not need to tally all the resources associated with an APC.

This becomes even more beneficial when multiple units use an assembly. For example, the Scout Hvy PLT US unit uses M3A3 assemblies. The Tank CO US uses M1A2 assemblies. The AR CAV TRP US uses M1A2s, M3A3s, and M1064 assemblies. While it might be a manageable process to configure M1A2s for one unit and M3A3s for another unit, it becomes tedious and error-prone to use them in multiple units and have to calculate the sum of all of their weapons and ammunition each time. Using assemblies makes it easy to assign multiple units of each type to a unit.

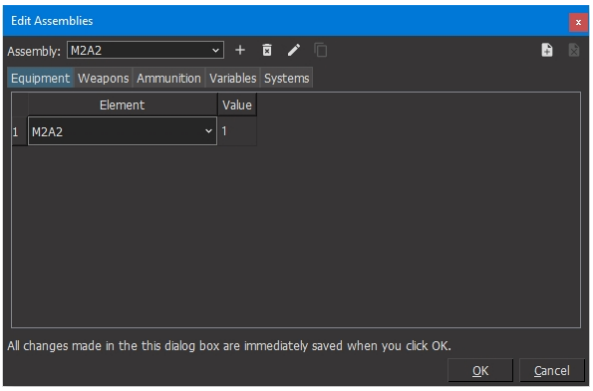
72.4.1 Creating Assemblies

When you create an assembly, you specify the equipment, weapons, ammunition, variables, and systems that are configured for a simulation object when you add the assembly to it.

➤ **To create an assembly:**

1. Choose **Edit > Assemblies**. The Edit Assemblies dialog box opens (Figure 600).

Figure 600. Edit Assemblies dialog box



2. Click the Add button (+). The New Assembly dialog box opens.
3. Enter a name for the assembly.
4. Optionally, select an assembly to base the new one on.
5. Click OK. The new assembly is added to the Assembly list.
6. Edit the assembly.

72.4.2 Copying an Assembly to Your SMS

When you create a new SMS by extending or including an existing one, the assemblies from the included SMS are read-only. If you want to customize an assembly, you can copy it from the included SMS to your new SMS and make changes. When you load your SMS, the customized assembly overwrites the one from the included SMS.

➤ **To copy an assembly:**

1. Choose **Edit > Assemblies**. The Edit Assemblies dialog box opens (Figure 600 above).
2. In the Assembly list, select the assembly that you want to copy from the included SMS.
3. Click the Copy button (). The assembly is copied to your SMS.
4. Edit the assembly.

If you delete an assembly that you copied, your SMS returns to using the one from the included SMS.

72.4.3 Editing Assemblies

An assembly specifies equipment, weapons, ammunition, variables, and systems. Variables represent entity resources that do not fit into weapons, ammunition, and equipment.

These include tangible resources, such as fuel, water, and personnel, and also intangible variables, such as health and combat power. If you do not specify particular variables for an assembly, you can specify them directly for a simulation object.

When you create an assembly, you can edit the properties of its systems. You can also edit system properties when you edit a simulation object. Many system properties do not change when you roll up a unit, but some do. For example, indirect fire weapons have a maximum sheaf parameter. The default maximum sheaf value for the 155 mm howitzer is 80 meters. If an artillery unit has 6 howitzers and you roll it up, the maximum sheaf value for the howitzers in that entity becomes 480.

> ⚠ If you specify variables for an assembly, when you roll up assemblies, the variables in the assembly overwrite the values that have been set directly. Therefore, it is recommended that you use them consistently in all of your assemblies and that you not edit these values directly in a unit if you know that you will use assemblies that might overwrite them.

➤ **To edit an assembly:**

1. Choose **Edit > Assemblies**. The Edit Assemblies dialog box opens (Figure 600 on the previous page).
2. In the Assembly list, select the assembly that you want to edit.
3. For the Equipment, Weapons, Ammunition, and Variables tabs:
    a. Click the Add New Assembly Item button (). A new row is added to the list of resources on the tab. In the Element column, it lists the first resource in the list of available resources.
    b. From the Element list, select the resource you want to add to this assembly.
    c. Double-click the Value cell to make it editable.
    d. Enter the amount of this resource that you want to be in this assembly.
4. For the Systems tab, add a system as described in step 3. Instead of entering a number in the Value column to complete, edit the system's properties. To edit properties:
    a. Click the Edit Properties button (). A Properties dialog box opens.
    b. Set the property values as desired.
5. Click OK.

72.4.4 Creating an Assembly from a Unit

In addition to creating a new assembly from the Edit Assemblies dialog box (as described in "72.4.1 Creating Assemblies" on page 1424), you can use an existing unit as the basis for a new assembly. Once it exists as an assembly, you can add it to other units and edit it using the Edit Assemblies dialog box (Figure 600 on the previous page). After creating the assembly, you can also copy any changes to the unit and update the assembly but that does overwrite any changes you made to the assembly itself.

➤ **To create or update an assembly based on a unit:**

1. Create or edit the unit as you would any aggregate-level simulation object. See "72.1. Creating Objects for Aggregate-Level Modeling" on page 1418.
2. Click Copy As Assembly.
3. Click Yes to confirm that you want to create or update the new assembly.
4. Click OK in the message notifying you of the assembly keyword.
5. Click OK in the message notifying you that the process was successful. The assembly is now available in the Edit Assemblies dialog box.

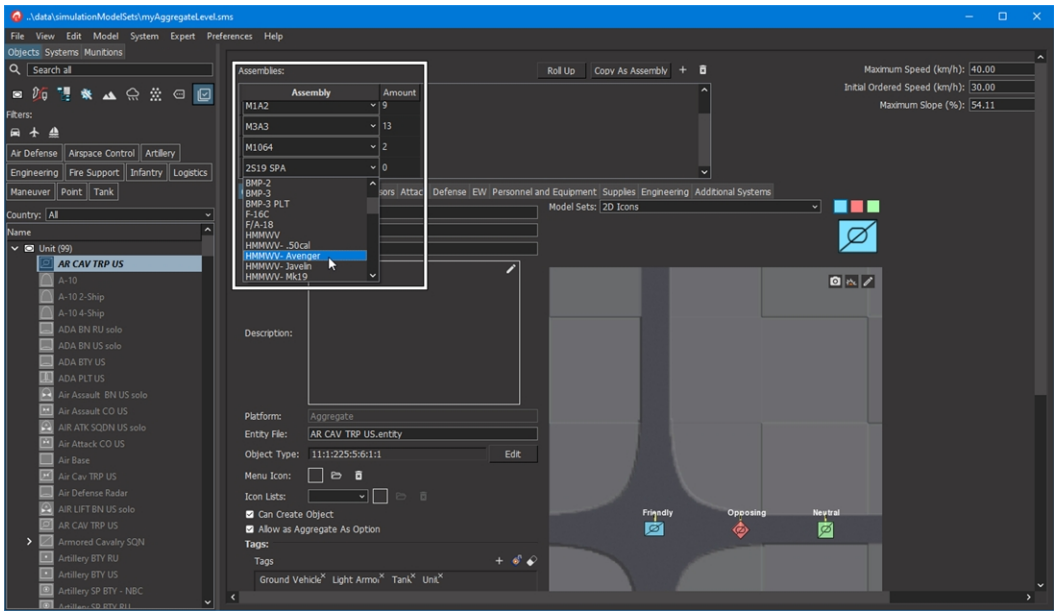
72.4.5 Adding Assemblies to a Unit

You can configure individual vehicles and weapons for a unit, or you can add assemblies, which add all the relevant parameters and combine them for the number of instances of each assembly that you add.

➤ **To add an assembly to a unit:**

1. Select the entity in the object list.
2. Click the Add button (+) above the Assemblies list (Figure 601). A new row is added to the list.

Figure 601. Assemblies table



3. In the Assembly column, select an assembly from the list.
4. Double-click the Amount value to make it editable.
5. Enter how many of this assembly that you want to add.

72.4.6 Removing Assemblies

➤ **To remove an assembly:**

1. In the Assemblies list, select the index number or any cell in the assembly you want to remove.
2. Click the Delete button ( ).

72.5. Rolling Up Assemblies

Once you assign a set of assemblies to a unit, you roll them up so that the full set of resources is applied to the unit. Assemblies are rolled up according to roll up rules.

> ⚠ When you roll up assemblies, the resources and modifiers in the assemblies overwrite the current values for the unit. If you have edited any of these resources or modifiers manually, your changes are lost. Any parameters that are not part of an assembly or roll up rules do not change.

➤ **To roll up assemblies for a unit:**

1. Select the unit.
2. Choose **Edit > Roll Up All** or, in the Assemblies area of the Simulation Object Editor window, click Roll Up. You are prompted to confirm the command.
3. Click Yes.

72.5.1 Creating Roll Up Rules

VR-Forces rolls up assemblies based on a set of rules. Roll up rules are not specific to simulation objects; they are global for an SMS. A roll up rule specifies the assemblies to roll up and the formula to use to roll them up:

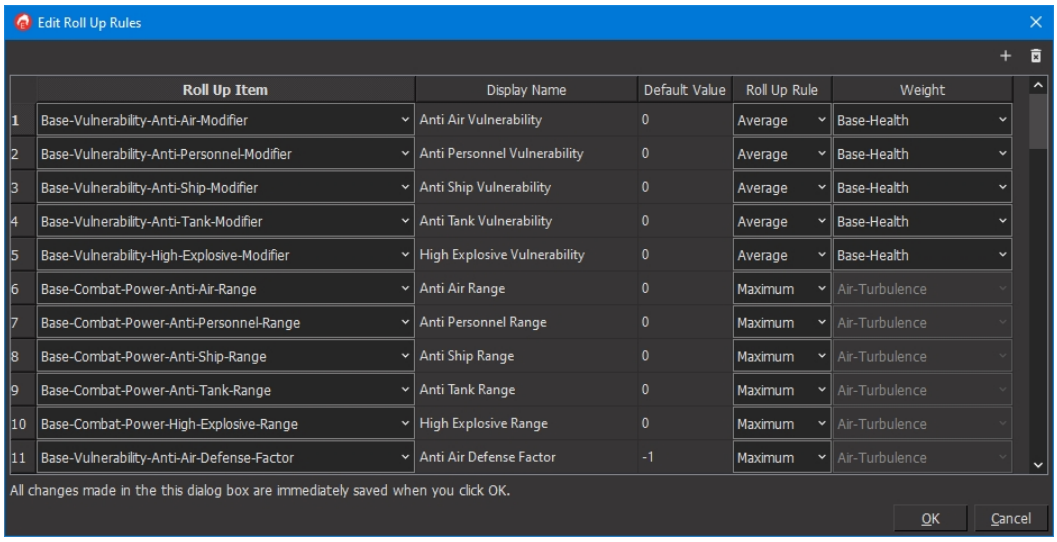
- Sum. Add up the values for all the assemblies.
- Average. Use the average value for identical variables in the assemblies.
- Maximum. Use the highest value specified for identical variables in the assemblies.
- Minimum. Use the lowest value specified for identical variables in the assemblies.

> ⓘ Ammunition, weapons, equipment, and systems are all rolled up by summation. There are no user-editable roll up rules for these resources.

➤ **To create roll up rules:**

1. Choose **Edit > Roll Up Rules**. The Edit Roll Up Rules dialog box opens (Figure 602).

Figure 602. Edit Roll Up Rules dialog box



2. Click the Add button (+). A new row is added to the list.
3. Select the type of roll up item from the list.
4. Specify a Display Name.
5. Specify a default value.
6. Select an option in the Roll Up Rule list.
7. If you are adding an Average roll up type rule, select an option in the Weight list. This creates a weighted average based on the item you want to give greater weight, such as health or vulnerability.
8. Add additional items as desired.
9. Click OK.

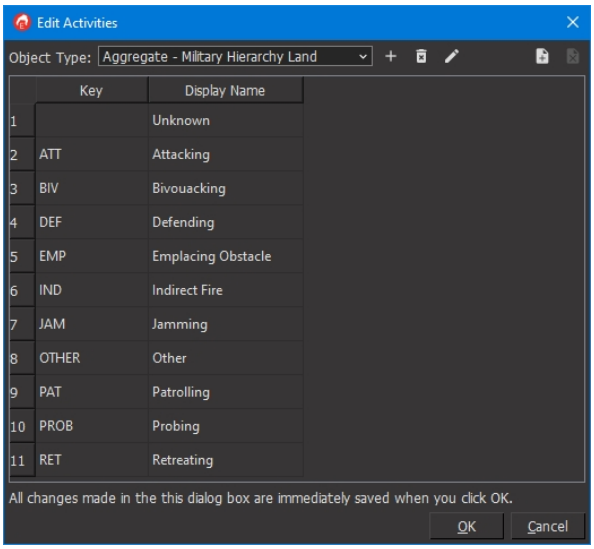
72.6. Adding Activities

You can assign simulation objects an activity. The Simulation Object Editor lets you add activities. When you add an activity, it is included in the list of activities for the Activity set data request. However, unless you add code or a Lua script that takes advantage of the new activity setting, assigning an activity to a simulation object will have no effect.

➤ **To add an activity to the Activities list:**

1. Choose **Edit > Activities**. The Edit Activities dialog box opens (Figure 603).

Figure 603. Edit Activities dialog box



2. Select an object type in the Object Type list, or click the Add button (+) to add a new type.
3. Click the Add Activity button (). A new entry is added to the list of activities. The Key cell is active for editing.
4. Enter a key value for the activity. This is the value that the code or Lua script uses to identify the activity.
5. Press **Tab** or double-click in the Display Name cell to make it editable.
6. Enter the text to display in the Activities dialog box.
7. Click OK.

72.7. Object Parameters for Aggregate-Level Modeling

The aggregate warfare model is data driven. It relies on the comparative strengths of opponents given their resources and movement posture. The data used is from an extensive set of parameters that are configured for each simulation object in the Simulation Object Editor.

Most parameters are in one of these categories:

- Real-world absolute values, such as liters of diesel fuel, number of personnel, and amount of equipment and ammunition.
- Modifiers. Modifiers change the base values for the various indicated conditions, such as posture, MOPP level, or footprint. All modifiers are multipliers. For example, speed modifiers are applied to the maximum speed. If the maximum speed for a simulation object is 20 km/hr and the modifier for overlapping another unit's

footprint is .9, the entity will not be able to go faster than 18 km/hr while it overlaps another unit.

- Systems. Movement system, sensor systems, other systems.
- Abstract values. Strength and health are abstract values. They represent the relative ability of simulation objects to damage other simulation objects and absorb damage.

For the simulation objects in aggregate-level SMSs provided with VR-Forces, real-world values are based on publicly available information about the types of simulation objects. The various modifiers applied to simulation object behavior are derived from customer requirements for the aggregate warfare model. Modifier values have been set from experimentation to provide what we believe to be realistic results. Values for health and strength have also been developed on a relative scale to obtain reasonable results.

If you add new simulation objects to an aggregate-level SMS, it is important that you try to set health and strength to values that fit with the existing simulation objects. If you create an entirely new SMS for aggregate-level scenarios, you are responsible for developing your own consistent way to represent the relative strengths of the simulation objects.

Parameters are organized in a set of tabbed pages. The following sections describe the parameters on each tab. The Additional Systems tab does not include aggregate warfare parameters. Use it to specify systems that did not logically belong on the other tabs.

72.7.1 General Parameters

Use the General tab to specify identifying information for a unit, such as its name, enumeration, tags, and model.

72.7.2 Physical Parameters

Physical parameters describe a simulation object's footprint and how it faces the external world for each posture, which is a simulation object's overall mode of interaction with its environment. Air units and ground units have different parameters.

Table 58. Physical parameters for ground units

Parameter	Description
Health	A quantification of the overall health of the entity.
Physical Footprint	The area notionally occupied by the simulation object. The footprint can vary for different postures based on the Footprint Modifier.
Height	Height of the physical footprint.
Footprint Modifier	The amount by which the footprint changes for the indicated posture.
Front Sector Size	The number of degrees, centered on the heading, within which forward-facing sensors and interactions take place.
Rear Sector Size	The number of degrees, centered on the heading -180º, within which rear-facing sensors and interactions take place.
Flank Sector Size	The number of degrees, centered on the heading +/- 90º, within which side-facing sensors and interactions take place.

Table 59. Physical parameters for air units

Parameter	Description
Aggregated Unit Movement	The movement system used by the unit.
Can Embark	Specifies whether the unit can embark on another unit.
Can Be Embarked Upon	Specifies whether other units can embark on this unit.
Health	The health value for this unit.
Physical Footprint	The area notionally occupied by the simulation object.
Mass	The weight of the unit.
Runway Length Required for Takeoff	The runway length this unit needs to take off.
Runway Length Required for Landing	The runway length this unit needs to land.
Runway Length Required for Landing with Drag Chute	The runway length required if this unit is landing with a drag chute deployed.
Has Drag Chute by Default	Whether this unit type should be equipped with a drag chute when it is created.

72.7.3 Movement Parameters

A unit has a movement system. You can specify the movement system on this tab and edit its parameters. You can also specify posture transition times.

Aircraft units do not have a Movement tab.

Movement can be affected by:

- Environmental factors. These properties are specified per movement system.
- Posture. A simulation object's posture can affect its speed.
- MOPP level transition time. The amount of time it takes for a unit to change from MOPP level 0 to one of the other MOPP levels.

The Movement tab also has settings for embarkation. These settings are the same as for simulation objects in an entity-level scenario. For details, see "Chapter 73. Configuring Embarkation" on page 1441.

Table 60 describes the movement parameters.

Table 60. Movement parameters

Parameter	Description
Max Speed Modifier When Overlapping Another Unit	The amount to modify a simulation object's maximum speed when its footprint overlaps another simulation object's footprint.
Max Speed Modifier (Posture)	The amount to modify the maximum speed based on the posture.
Health Percent For Rout	The amount to modify health when in the rout posture.
Transition Time from Level 0	The amount of time it takes to transition from MOPP Level 0 to a higher MOPP level.
Max Speed Modifier (MOPP Level)	The amount to modify the maximum speed based on the MOPP Level.

Editing Movement System Properties

Table 61 describes movement system properties. Systems do not all have the same properties.

Table 61. Movement system properties

Property	Description
Rain Modifier by Intensity	The amount speed is affected by the intensity of rain. The lower the modifier value, the more slowly the simulation object moves as rain intensity increases.
Snow Modifier by Intensity	The amount speed is affected by the intensity of snow. The lower the modifier value, the more slowly the simulation object moves as snow intensity increases.
Visibility Degrades Max Speed When Below	If visibility drops below this value, maximum speed is reduced.
Maximum Amount Visibility Can Degrade Speed	The maximum amount that speed decreases as visibility decreases. Higher values decrease speed more.
Wind Degrades Max Speed When Above	If wind speed exceeds this value, the maximum speed of the entity is reduced. (Aircraft only.)
Wind Degrades Until	Degrade speed due to wind until the wind speed reaches this value. Then degrade speed at the maximum rate. (Aircraft only.)
Maximum Amount Wind Can Degrade Speed	The maximum amount that wind can degrade speed. (Aircraft only.)
Retrograde Movement Penalty	Speed is reduced by this amount when moving in retrograde. (Ground movement only.)

➤ **To edit movement system properties:**

1. Click the Edit Properties button ( ). The Edit Properties dialog box opens.
2. Edit the values for the various properties.
3. Click OK.

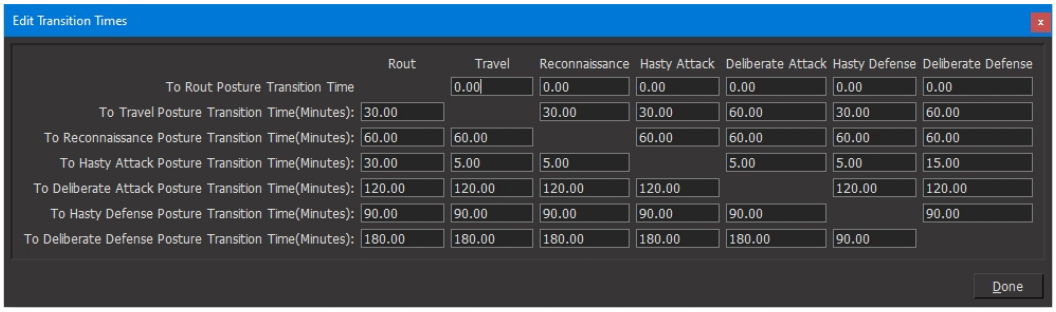
Editing Posture Transition Times

VR-Forces models the amount of time it takes for a unit to transition from one posture to another. You can specify the transition times for each movement system.

➤ **To edit posture transition times:**

1. Select the Movement tab.
2. Click Edit Transition Times. The Edit Transition Times dialog box opens (Figure 604).

Figure 604. Edit Transition Times dialog box



3. Change the times, in minutes, as desired. Changes take effect immediately.
4. Click Done.

72.7.4 Sensor Parameters

Use the Sensors tab to specify the sensors that a simulation object has. You can add and remove sensor systems and set their properties using the standard Simulation Object Editor procedures. (For details, see "70.4. Editing a Simulation Object's Systems" on page 1390.)

You can specify the simulation object's sensor signatures by choosing from a preconfigured set of signatures or by defining a custom set. The Sensor Signatures parameters specify the distance at which other simulation objects can sense the simulation object you are editing using the different types of sensors. The preconfigured signatures make it easy to consistently apply the same set of signatures to similar object types.

As with movement, a unit's posture and MOPP level affect its ability to use its sensors. You can specify signature modifiers for each posture and MOPP Level. (Posture and MOPP level modifiers do not apply to air units.)

72.7.5 Attack Parameters

Simulation objects have combat power in five domains: Anti-Air, Anti-Tank, High Explosive, Anti-Personnel, and Anti-Ship. The Attack tab specifies the parameters that determine a unit's ability to attack an opponent in these domains. It allows you to add weapon systems to the simulation object.

Aircraft units do not have an Attack tab.

Table 62. Attack parameters

Parameter	Description
Range	The distance at which a simulation object can attack in each domain.
Hit Factor	When compared to the defense factor for a target simulation object, determines the probability of the attack hitting the target.
Strength	The simulation object's strength in each type of combat. This is an abstract value.
Ammunition Usage	The number of rounds per minute used for each type of combat. This value is apportioned among ammunition of the same type. For example, if the ammunition usage for Anti-Personnel combat is 1000 rounds per minute and a simulation object has 12.7 mm ammunition and 7.62 mm ammunition, both of which are type Anti-Personnel, then the simulation object uses 500 rounds per minute of each type of ammunition.
Ammunition Auto-Resupply Rate	The amount of ammunition that is automatically resupplied each day if auto-resupply is enabled for this simulation object.
Ammunition	The types of ammunition this simulation object has, the amount of each type, and whether pacing or tracking is enabled.
Distribution	The proportion of casualty types from an attack.
Sector Combat Power Modifiers	The amount the base combat power is modified depending on which sector is being attacked.
Posture Combat Power Modifiers	The effect on combat power of each posture.
Combat Range Modifiers	The effect on the range at which a simulation object can attack based on its posture.
CID Level Combat Power Modifiers	The effect on combat power of the enemy detection and classification level.
MOPP Level Combat Power Modifiers	The effect on combat power of the MOPP Level.

Table 62. Attack parameters (continued)

Parameter	Description
If Morale is Less Than	A percentage of morale below which combat power is affected.
Adjust Combat Power By	The combat power modifier to apply if morale is less than the specified amount.
Environmental Conditions Combat Power Modifier	The combat power modifier to apply if precipitation intensity is at its maximum or if there is no illumination.
Weapon System Auto-Resupply Rate	The number of rounds per day supplied to each weapon system if auto-resupply is enabled.
Weapon System Attack Interval	The amount of time between each weapon system attack by this unit.

72.7.6 Defense Parameters

Use the Defense tab to specify modifiers to a simulation object's vulnerability. Vulnerability is specified within each weapon domain.

Aircraft units do not have a Defense tab.

Table 63. Defense parameters

Parameter	Description
Vulnerability	The amount that a simulation object is vulnerable to the different types of attack.
Defense Factor for Guided Munitions	The ability to defend against guided munitions.
NBC Vulnerability	Vulnerability to nuclear, biological, or chemical contamination.
Posture Vulnerability Modifier	The effect of different postures on the simulation object's ability to defend itself.
Sector Vulnerability Modifier	The ability of the simulation object to defend itself to the front, rear, and flanks.
MOPP Level Vulnerability Modifier	The ability of the simulation object to defend against NBC contamination at different MOPP Levels.
Health Percent for Firepower Kill	The level of health below which this unit will not be able to fire.
Health Percent for Mobility Kill	The level of health below which this unit will not be able to move.
Health Percent for Sensor Kill	The level of health below which this unit will not be able to sensor other objects.

72.7.7 Combat Parameters

Air units have a Combat tab. This tab lets you configure weapon systems, combat systems, loadout, and health parameters.

Table 64. Combat parameters

Parameter	Description
Weapon Systems	Add, remove, and edit properties for the weapon systems this unit has.
Combat Systems	Add, remove, and edit properties for other combat-related systems.
Loadout	The type of loadout this unit uses. You can specify a default by clicking the star, and then lists showing the loadouts indicate the default with a star. (Loadouts are specific to each unit. There is no library of preconfigured loadouts available to select from.)
Loadout Items	The specific munitions in the loadout.
Distribution	The percent chance that a casualty caused by this unit will be of a specific casualty type.
Defense Factor for Guided Munitions	The defense factor to apply when this unit is struck by a guided munition.
Weapon System Attack Interval	The amount of time between each weapon system attack by this unit.
Health Percent for Firepower Kill	The level of health below which this unit will not be able to fire.
Health Percent for Mobility Kill	The level of health below which this unit will not be able to move.
Health Percent for Sensor Kill	The level of health below which this unit will not be able to sensor other objects.

72.7.8 Electronic Warfare (EW) Parameters

The EW tab lets you specify a simulation object's ability to defend against electronic warfare and how much it depends on communications. You can add EW systems.

72.7.9 Personnel and Equipment Parameters

Use the Personnel and Equipment tab to specify the order of battle for the simulation object. These values are not used by the sim engine in the simulation. They are for reporting purposes. The configured personnel and equipment amounts are multiplied by the simulation object's health to arrive at the current amount.

Table 65. Personnel and Equipment parameters

Parameter	Description
Personnel Officers	Number of officers.
Personnel WOs	Number of warrant officers.
Personnel NCOs	Number of non-commissioned officers.
Personnel Enlisted	Number of enlisted persons.
Equipment	List of equipment types, the number of each, and whether it is flagged for pacing, tracking, or none.
Weapons	List of weapon types, the number of each, and whether it is flagged for pacing, tracking, or none. (Ground units only.)

72.7.10 Supplies Parameters

For ground units, the Supplies tab allows you to specify the amount, use per day, use per kilometer, pacing/tracking, and auto-resupply rate for these supplies:

- Food.
- Water.
- Oil.
- Lubricant.
- Motor gas.
- Aviation fuel.
- Diesel fuel.
- Other Supplies. You can add any other type of supply that you want. The list of items in the Supply drop-down list is drawn from *./data/simulationModelSets/AggregateLevel/gui/rwKeywordValuePairs.xml*. (For information about this file, see "72.9. Editing Reader/Writer Key/Value Pairs" on page 1440.) You can enter in any other supply type that you want to.

  The supplies added here are only used by engineering objects. To use a supply, you configure it here and in the engineering object. For example, a combat engineering company has other supplies called Construction-Material and Explosives.

You also use this tab to specify the conditions under which a simulation object can receive supplies.

For air units, you can specify the amount of aviation fuel, the aerial refueling system, and other supplies.

72.8. Configuring Combat Engineering Objects

Combat engineering objects (CEOs) can affect mobility of simulation objects and can cause attrition. Therefore, their simulation models include the information needed for these effects. Since CEOs can be constructed by units, their models include parameters that specify the types of materials needed to create them and sizing information.

The actual process of configuring CEOs is similar to that of other objects. Table 66 describes the parameters specific to CEOs. Some parameters do not apply to all CEOs.

Table 66. Combat engineering object parameters

Parameter	Description
Short Name	The name displayed on the map and sent as marking text over the network.
Height	The height of the object.
Length	Maximum length of single constructed objects. If a longer object is specified, it is created as several objects.
Width	The width of a line object.
Minimum Completion Percentage for Effect	The minimum completion percentage before the object affects simulation objects.
Minimum Completion Percentage to Detect	The minimum completion percentage before simulation objects can detect the object.
Attrition	The rate of attrition to an affected simulation object in each domain.
Mobility Modifiers	The amount that mobility of an affected simulation object is modified in each type of movement.
Vulnerability Modifiers	The amount that an affected simulation object's vulnerability is modified in each domain.
Signature Modifiers	For object types that can be affected by this object, the amount that each type of sensor signature is affected.
Resource Required	The type of resource needed to construct this object.
Resource Amount	The amount of the required resource needed to build this object.

Table 66. Combat engineering object parameters (continued)

Parameter	Description
Is Concealed	If selected, the object is detected based on a random probability in the detecting sensor. If cleared, the object is always detected by the appropriate sensor.
Maximum Relative Size	For areal objects, the maximum size of a simulation object, relative to the object, for it to be affected by the object. For example, if the maximum relative size for a fortified area is .75, a simulation object must be 75% of the size of the area or smaller for it to be protected.

72.9. Editing Reader/Writer Key/Value Pairs

Some display text is not automatically included in the *untranslated.ts* file provided with VR-Forces. The Simulation Object Editor lets you map some internal keywords to display text so that you can translate it. These internal keywords are called reader/writer keys and they are managed using key/value pairs. These mappings are stored in *./data/simulationModelSets/AggregateLevel/gui/rwKeywordValuePairs.xml*. If you edit this file, do not change the **paramName**; change the string mapped to it.

Besides allowing for translation, you can add new entries to this file. Table 67 describes the effect of adding new keyword:value pairs.

Table 67. Effect of editing rwKeywordValuePairs.xml parameters

Parameter	Result
Type	Added to the Type list for the Edit Ammunition and Equipment dialog box, Ammunition tab, in the Simulation Object Editor.
Category	Added to the Category list for the Edit Ammunition and Equipment dialog box, Weapon tab and Equipment tab, in the Simulation Object Editor.
Posture	Added to the list of Postures in the Posture set data request. However, selecting such a posture would not have any effect unless complementary code is written to update the simulation engine.
Resource-Item	Added to the Supply list for configuring Other Supplies on the Supplies tab for simulation objects in the Simulation Object Editor.



73. Configuring Embarkation

This chapter explains how to configure entities for embarkation. It includes a tutorial for configuring embarkation slots.

73.1. Enabling Embarkation 1442
73.2. Adjusting the View While Configuring Embarkation 1443
73.3. Configuring Slots 1444
73.3.1 Adding Slots 1444
73.3.2 Configuring Slot Parameters 1446
73.3.3 Configuring Slot Exclusions 1449
73.3.4 Specifying Ingress and Egress Techniques 1450
73.4. Adding Ingress and Egress Points 1451
73.5. Embarkation Slot Tutorial 1453
73.5.1 Configure the Super Tanker for Embarkation 1454
73.5.2 Configure Slots for the Super Tanker 1456
73.5.3 Add Ingress Points 1459
73.5.4 Add Egress Points 1460
73.5.5 Save the New Configuration 1461
73.5.6 Test the Embarkation Configuration 1461

73.1. Enabling Embarkation

If an entity allows other entities to embark on it, you can configure:

- Allowed entity types. The types of entities that can embark.
- Slots. The location on an entity where embarked entities move to. The number of slots on an entity controls how many entities can embark on it using the Embark task. For entities that are visible when they are embarked, such as humans on a truck or aircraft on an aircraft carrier, it is important to specify sensible-looking slots.

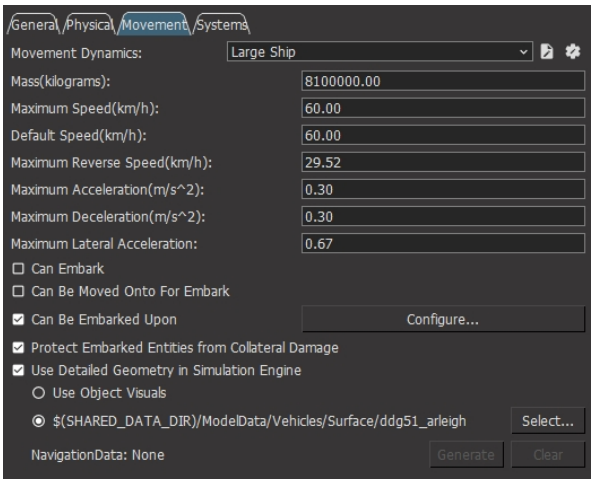
  Slots can have a type and a name. Use of slot types and names allows you to tell a simulation object to embark in a specific slot.
- Ingress points. The locations on an entity where entities embark before moving to a slot.
- Egress points. The locations on an entity from which entities disembark.

> Units can support embarkation, but the usual use of embarkation is for entities to embark on other entities. In entity-level scenarios, if a unit embarks, the individual members of the unit carry out the embarkation task.

➤ **To enable embarkation for an entity:**

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. Select the Movement tab (Figure 605).

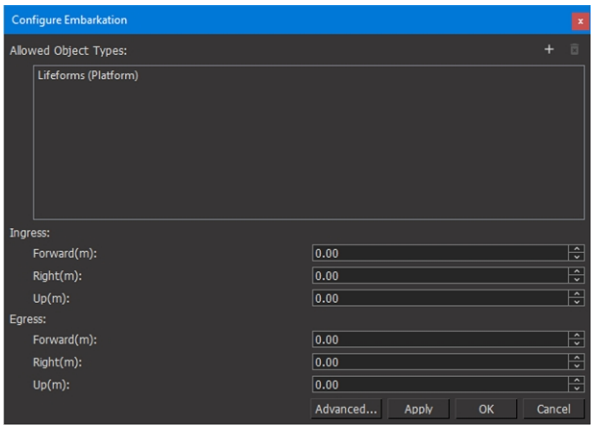
Figure 605. Movement tab



3. If Can Be Embarked Upon is not selected, select it.
4. Optionally, select Can Be Moved Onto For Embark. This option lets ground-clamped entities identify entities that they can embark on. They can then walk or drive onto the entity and automatically become embarked.

5. Optionally, select Protect Embarked Entities from Collateral Damage. If selected, embarked entities are not damaged if the parent entity sustains collateral damage.
6. Click Configure. The Configure Embarkation dialog box opens (Figure 606).

Figure 606. Configure Embarkation dialog box



7. Click Add. The Add Type dialog box opens.
8. Select a type of entity that can embark.
9. Optionally, add additional entity types.
10. For each entity type listed, specify ingress and egress points. Values are in meters relative to the center of the parent entity. For more information, see "73.4. Adding Ingress and Egress Points" on page 1451.
11. Optionally, configure slots. For more information, see "73.3. Configuring Slots" on the next page.
12. Click OK.

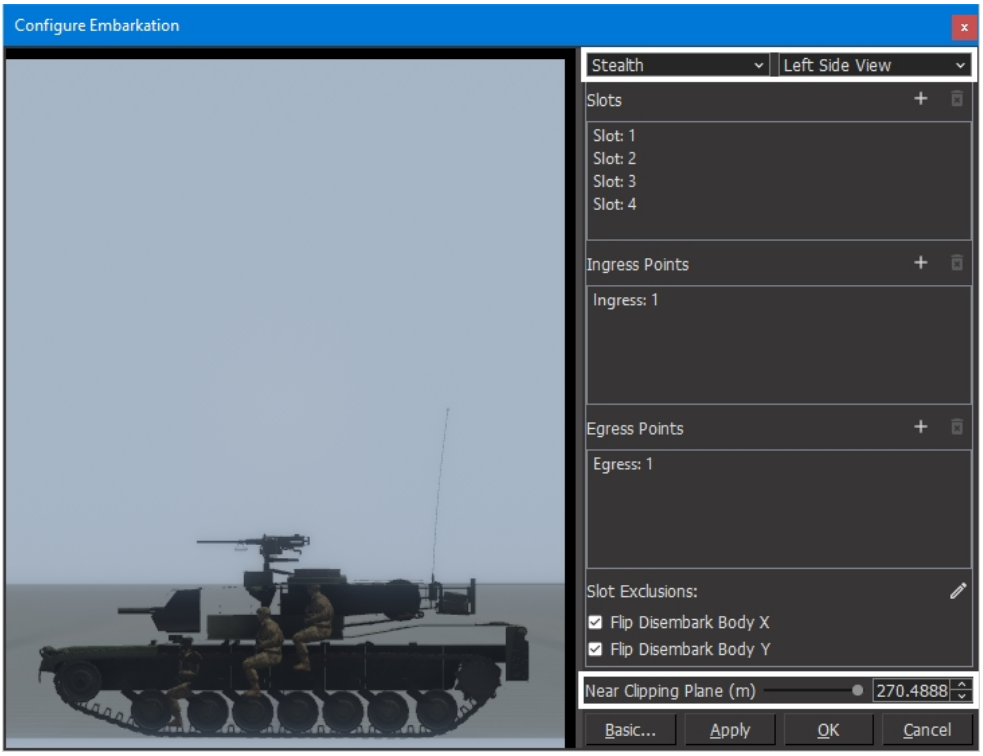
73.2. Adjusting the View While Configuring Embarkation

You can adjust the perspective of the Configure Embarkation dialog box, allowing you to view the entity as needed to define embarkation slots as well as ingress and egress points.

You are able to:

- Change the view mode to either Plan View (2D) or Stealth (3D).
- Select one of the views for the Stealth view mode, such as Top Down or Left Side View.
- Use the mouse or keyboard to zoom in or out or otherwise change your view of the entity.
- Adjust the Near Clipping Plane to be able to see the inside when you are defining slots within the entity.

Figure 607. M1A2 Abrams MBT with view for configuring embarkation



73.3. Configuring Slots

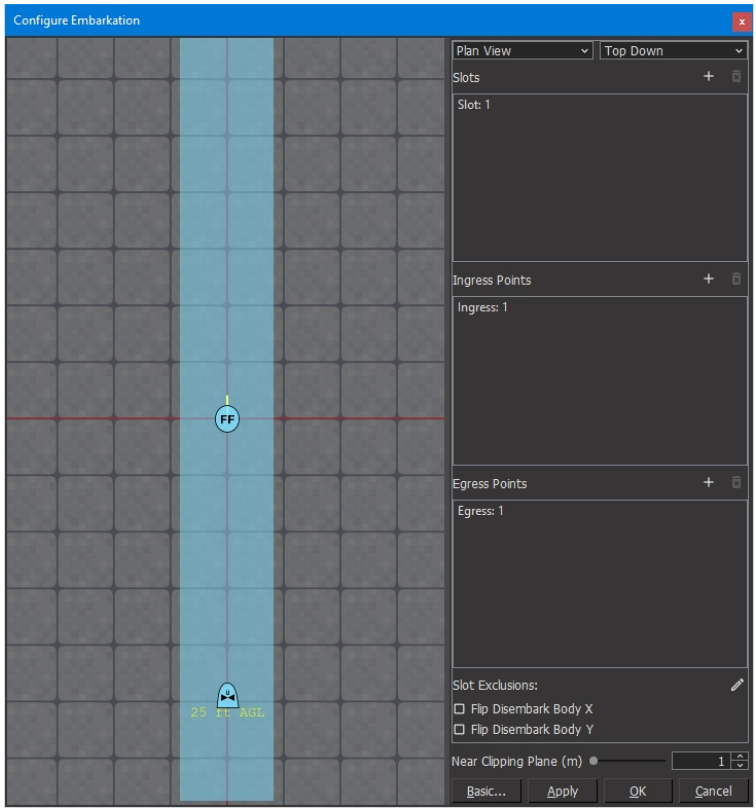
In the simple embarkation configuration process, VR-Forces adds one slot, located at the center of the parent entity. You can edit that slot. You can also add additional slots if you want embarked entities to be in specific locations on the parent.

73.3.1 Adding Slots

➤ **To add a slot:**

1. In the Simulation Object Editor object list, select the simulation object that you want to configure embarkation for.
2. Select the Movement tab (Figure 605 on page 1442).
3. Select the Can Be Embarked Upon check box.
4. Click Configure. The Configure Embarkation dialog box opens (Figure 606 on the previous page).
5. If you have not already specified the simulation object types that can embark, specify a simulation object type as described in "73.1. Enabling Embarkation" on page 1442.
6. Click Advanced. The dialog box redisplays. It shows a 2D view of the bounding volume of the simulation object for which you are configuring embarkation (Figure 608).

Figure 608. Configure Embarkation Advanced



7. Optionally, change the view to see the entity in a way that best allows you to see the slot positions. See "73.2. Adjusting the View While Configuring Embarkation" on page 1443.
8. Click the Add button above the Slots window (Figure 608). A slot is added to the list and the dialog box redisplays to show the slot's parameters (Figure 609).

Figure 609. Slot parameters



9. Optionally, configure Slot Exclusions as described in "73.3.3 Configuring Slot Exclusions" on page 1449.
10. Optionally, configure other slot parameters as described in "73.3.2 Configuring Slot Parameters" below.

73.3.2 Configuring Slot Parameters

You can specify these parameters for a slot:

- The object type that can use it. You can configure different object types in different slots on a simulation object.

  For each field in the object type enumeration, you can specify more than one value. For example, the category value for a helicopter can be 20-25 for military helicopters, 50 for unmanned, and 90-92 for civilian. If you just want to allow a specific category for a given slot, specify that category. If you want to specify a range of values, separate them with a hyphen, for example, 21-24. To specify several specific categories, separate them with commas, for example, 21, 25, 50. You can also combine these options. To specify all helicopters, you could set the object type to 1:2:-1:20-25,50,90-92:-1:-1:-1.
- A name for the slot.
- A slot type.

- The DIS station name. This allows you to indicate the relative position of one entity to another by an identified position rather than an x,y,z offset.
- The position on the parent entity relative to the center of the entity's bounding volume.
- The orientation of the slot relative to the center of the entity.
- The appearance of the simulation object type when it is in the slot. For example, a human embarked on a truck might have its posture set to Sitting.
- For slots that accept humans, an animation to use while embarked, for the ingress, and for the egress. See "73.3.4 Specifying Ingress and Egress Techniques" on page 1450
- Whether the embarking entity should just jump to the slot location, rather than using normal entity movement. For example, an aircraft will taxi to a slot on a carrier deck.
- Whether the embarking entity should turn (snap) to the heading of the parent when the entity starts to move to the egress point or starts to move to the embarkation point. (Some movement models do not work well when snapping to a heading. For those models, use the Orient on Ingress option, not Turn to Heading.)
- Whether the embarking entity should smoothly orient (turn) to the heading of the parent when it reaches the embarkation point.
- Whether the embarking entity should be invisible while it is embarked.
- How many simulation objects the slot can hold.

➤ **To specify slot parameters:**

1. To change the side of a parent simulation object that the embarked simulation objects exit on, select the Flip Disembark Body X check box, the Flip Disembark Body Y check box, or both.
2. If you have added a new slot, in the Object Type field, specify the object enumeration for the type of simulation object that can use this slot. If this is a previously created slot, you can edit the object type. You can configure different object types in different slots on a simulation object.
3. If you want to give the slot a name, enter it in the Slot Name text box.
4. If you want to categorize this slot as a specific named type (a text string, not an object type enumeration), enter it in the Slot Type text box.
5. In the Position group box, specify the location of the slot relative to the center of the entity's bounding volume.
6. In the Orientation group box, specify the orientation of the slot relative to the center of the entity.
7. Optionally, specify the appearance of the simulation object type when it is in the slot. If you want to specify an animation for a human, specify a posture that supports animations, such as Upright, Standing Still. The default is to ignore the appearance.
8. Optionally, specify an animation that the entity will use when it embarks.

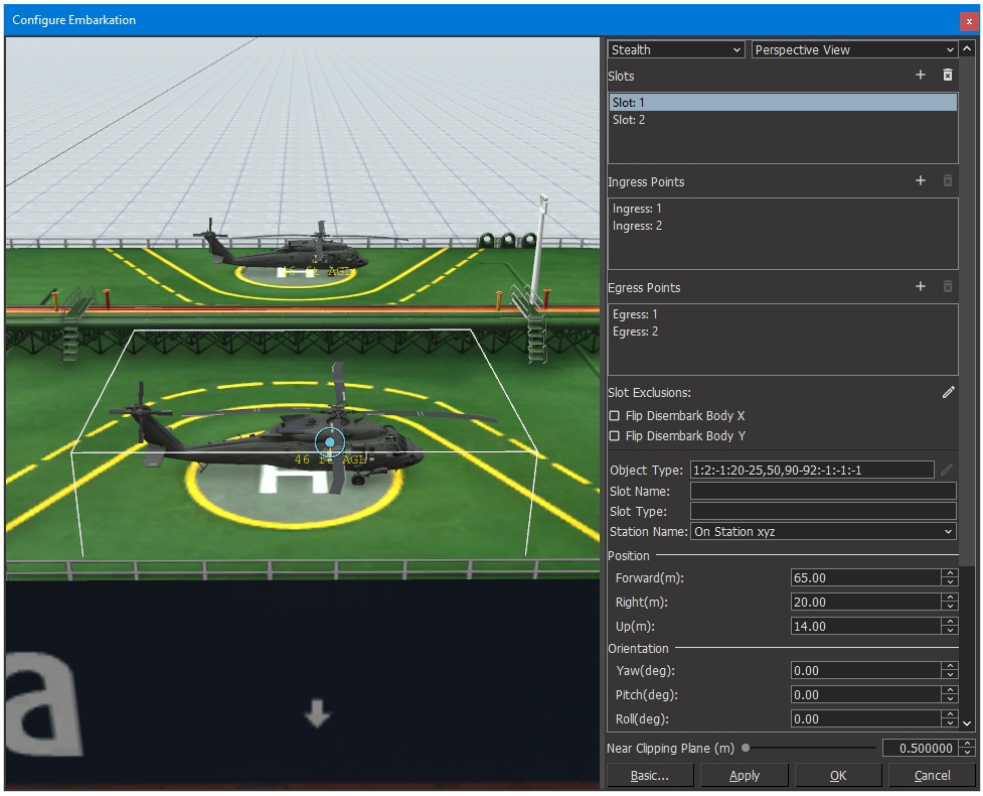
> Animations only apply to humans. The available animations depend on the entity's posture, as specified by the Appearance parameter.

9. Optionally, specify an ingress technique. For details about specifying an ingress technique, see "73.3.4 Specifying Ingress and Egress Techniques" on page 1450.
10. Optionally specify an egress technique. For details about specifying an egress technique, see "73.3.4 Specifying Ingress and Egress Techniques" on page 1450.
11. Optionally, for lifeforms only, if the Appearance specifies a posture that has animations, click the Edit button and select an animation.
12. If you want the embarked simulation object to immediately jump to the slot from the ingress point, select Jump To Location. If this check box is clear, the simulation object will move to the slot.
13. If you want the embarked simulation object to snap to the same heading as the parent entity, select Turn To Heading.
14. If you want the embarking entity to smoothly orient itself to the heading of the entity when it reaches the ingress point, select Orient on Ingress.
15. If you want the embarked simulation object to be invisible in 3D observer modes, select Invisible When Embarked.
16. In the Capacity text box, specify the number of entities that can use this slot. -1 means an unlimited number.

Previewing the Embarked Entity's Position

In the Stealth view of the Configure Embarkation dialog box, you can move the observer to see the position of the example embarked entity (Figure 610). This can help you position it where you want. Note that this is relative to the visual model; see "70.1.6 Adding Object Geometry to a Simulation Object" on page 1380. Testing it in a scenario is the best method for seeing whether it embarks as desired, as covered in the tutorial at the end of this chapter.

Figure 610. Side view of embarked entity



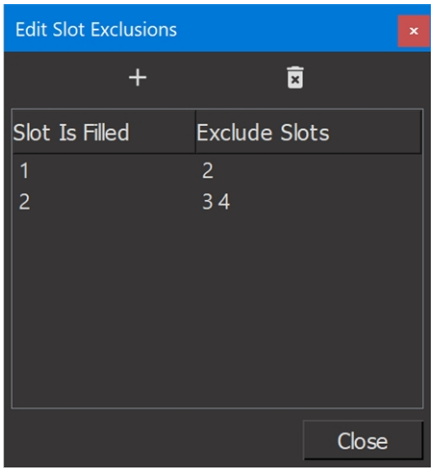
73.3.3 Configuring Slot Exclusions

A slot exclusion specifies that if a given slot is occupied, other slots cannot be occupied. For example, an entity might allow both tanks and dismounted infantry entities to embark. If no tanks are embarked, the entity can hold 20 DI-Guys. But if a tank is embarked, it uses up 10 slots and now only 10 DI-Guys can embark.

➤ **To configure slot exclusions:**

1. On the Slot Exclusions line, click Edit. The Edit Slot Exclusions dialog box opens (Figure 611).

Figure 611. Edit Slot Exclusions dialog box



2. Click the Add button (+). A blank line is added to the list.
3. Double-click the cell in the Slot Is Filled column. The cell becomes editable.
4. Enter the number of the slot that you want to configure.
5. Double-click the cell in the Exclude Slots column.
6. Enter the numbers of the slots that cannot accept entities if the slot you are configuring is filled. You can enter more than one slot to be excluded. Separate the slot numbers with a space.
7. Repeat this procedure to add additional exclusions. You can also edit existing exclusions and delete them.

73.3.4 Specifying Ingress and Egress Techniques

When you configure a slot, you can specify an ingress and egress technique. An ingress or egress technique associates a string with a script that runs when the entity embarks. The technique string is associated with a script ID in the OPE file for the entity type.

To specify a technique in an OPE file, add the **technique-map** block to the embarkation controller. The *Human.ope* file has this technique map:

```
(embarkation-controller
  ...
  (invisible-at-origin True)
  (technique-map
    (GetInCar "get_in_car")
    (GetOutOfCar "get_out_of_car")
  )
)
```

In this case, putting the string GetInCar in the Ingress parameter tells VR-Forces that when the entity embarks, it should run the script that has the script ID get_in_car.

VR-Forces includes two scripts, Get In Car and Get Out Of Car (in the Civilian Vehicle folder), that show how to write a script that implements an ingress or egress technique. As shown in the comments in these scripts, there is a set of standard task variables. Do not define the variables. The scripts gets their values from the embarkation code.

73.4. Adding Ingress and Egress Points

An ingress point specifies where an entity embarks onto another entity. An ingress point can specify a path that the entity follows to reach the ingress point and how it moves once it is on the entity. For example, a jet embarking on an aircraft carrier can taxi to its slot.

> If you do not specify an ingress point, entities will not move towards the parent before embarking. They will teleport to the slot.

An egress point specifies where an entity disembarks from the parent entity.

You can specify that an ingress point should only support some of the slots on the entity. You might want some types of entities embarking at one ingress point and moving to certain slots, while other entities embark at a different location and use other slots. For example, if you are simulating a ferry, cars and trucks would enter at the front or rear and move below deck. Foot passengers would enter at the sides and move to the passenger decks.

When you add an allowed entity type, the Simulation Object Editor automatically creates a slot for that entity type. The ingress and egress points that you configure for that entity type on the basic version of the Configure Embarkation dialog box are automatically associated with the same slot.

In Advanced mode, you can add additional ingress and egress points and associate them with whichever slot you want. You do not have to add them in pairs.

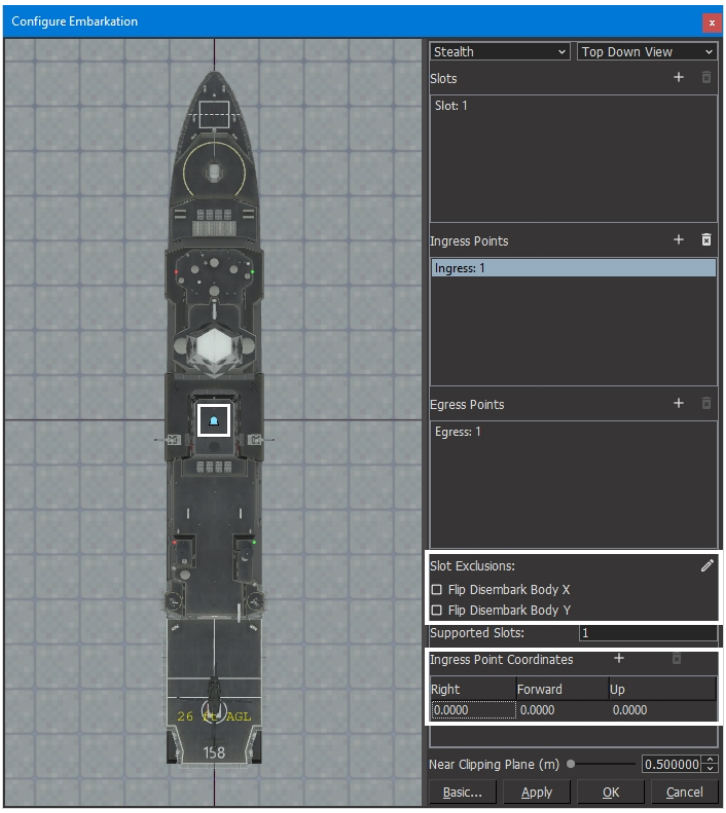
> This procedure describes configuring an ingress point. To configure an egress point, substitute "egress" for "ingress" in each step.

➤ **To add an ingress point:**

1. In the Simulation Object Editor object list, select the entity that you want to configure embarkation for.
2. Select the Movement tab (Figure 605 on page 1442).
3. Select the Can Be Embarked Upon check box.
4. Click Configure. The Configure Embarkation dialog box opens (Figure 606 on page 1443).
5. Click Advanced (if needed). The dialog box redisplays. It shows an entity icon in Plan View mode.
6. Optionally, change the view to see the entity in a way that best allows you to see the slot positions. See "73.2. Adjusting the View While Configuring Embarkation" on page 1443.

7. Click the Add button (+) above the Ingress Points list. An ingress point is added to the list. A vertex is added to the display window at the center of the entity bounding volume (Figure 612). The coordinates for the ingress point displays in the table near the bottom of the dialog box.

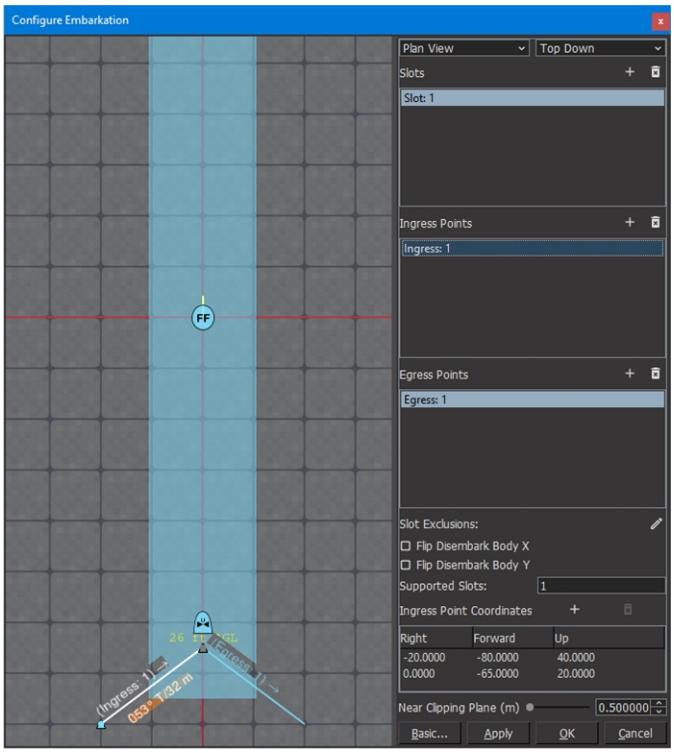
Figure 612. Ingress Point (Stealth View)



8. Double-click the Right value for the Ingress point to make it editable.
9. Enter the Right value.
10. Repeat for the Forward and Up values.
11. To specify a path for the embarking entity:
    a. Click the Add button for the Ingress Point coordinates list. A new coordinate is added.
    b. Specify the Right, Forward, and Up values for the new coordinate. The path displays in the window. Entities move along the points in the order they are shown in the table. In the Plan View, an arrow indicates the direction entities

will move along the path (Figure 613).

Figure 613. Ingress point configuration (Plan View)



12. Optionally, in the Supported Slots field, enter the numbers for the slots that should support this ingress point. If you leave the field blank, it supports all slots. To specify more than one slot, separate the numbers with a space.
13. To change the side of a parent simulation object that the embarked simulation objects exit on, select the Flip Disembark Body X check box, the Flip Disembark Body Y check box, or both (Figure 608 on page 1445).
14. Click OK.

73.5. Embarkation Slot Tutorial

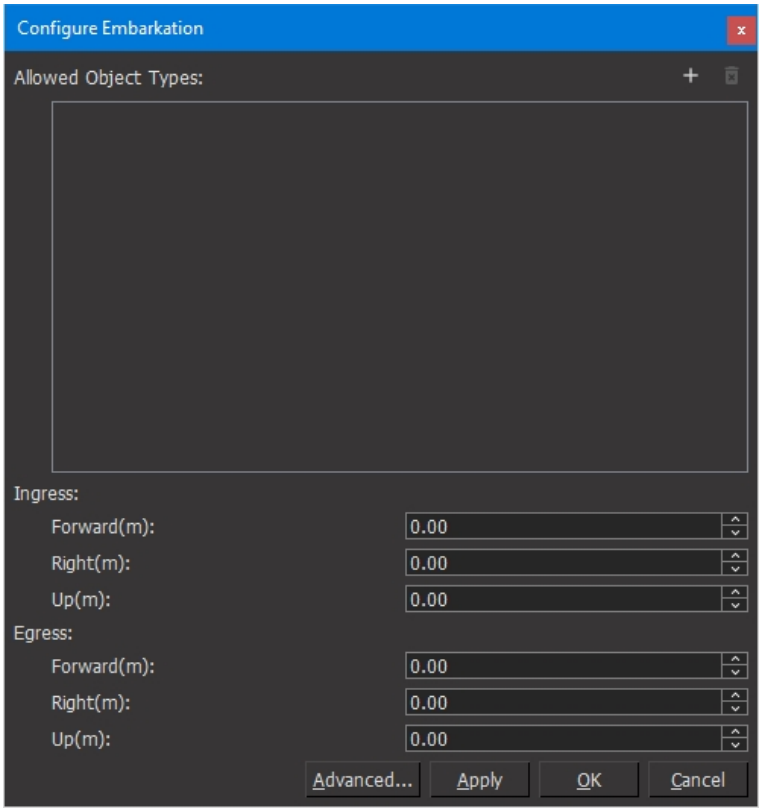
This tutorial shows how to configure an entity for embarkation. The super tanker does not have slots configured for embarkation in the default VR-Forces *EntityLevel.sms*. However, it has two helipads on the deck, which makes it ideal for embarkation. We will add embarkation slots, ingress points, and egress points to the super tanker. Then we will demonstrate the new slots in a scenario.

73.5.1 Configure the Super Tanker for Embarkation

➤ **To configure the super tanker for embarkation:**

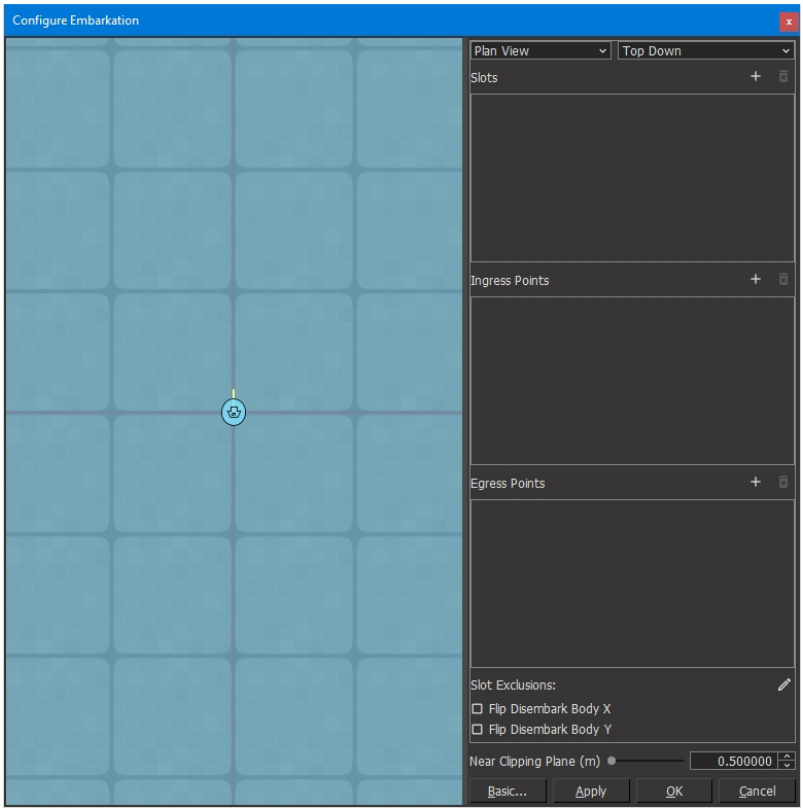
1. Start the Simulation Object Editor.
2. Choose **File > New Simulation Model Set**. The New Simulation Model Set dialog box opens.
3. Select the New Simulation Model Set option.
4. Click Next. The Simulation Model Set Type dialog box opens.
5. Select the Extend Existing Simulation Model Set option.
6. Click Next. The Extend Existing Simulation Model Set dialog box opens.
7. Select *EntityLevel.sms*.
8. Click Next. The New Model Set Name dialog box opens.
9. Enter a name for the new SMS.
10. Click Finish. The new SMS is created and loaded.
11. In the list of entities, select Super Tanker. It is read-only because it is included from *EntityLevel.sms*.
12. At the top of the Simulation Object Editor window, click Promote. You can now edit the Super Tanker.
13. Select the Movement tab (Figure 605 on page 1442).
14. Select the Can Be Embarked Upon check box.
15. Click Configure. The Configure Embarkation dialog box opens (Figure 614).

Figure 614. Configure Embarkation dialog box



16. Click Add (+). The Add Type dialog box opens.
17. In the list of entity types, select Rotary Wings (Platform) and then click OK. Rotary Wings (Platform) is listed in the Allowed Object Types window.
18. Click Advanced. The Configure Embarkation dialog box redisplays (Figure 615). We are now ready to configure slots.

Figure 615. Configure Embarkation 2D



73.5.2 Configure Slots for the Super Tanker

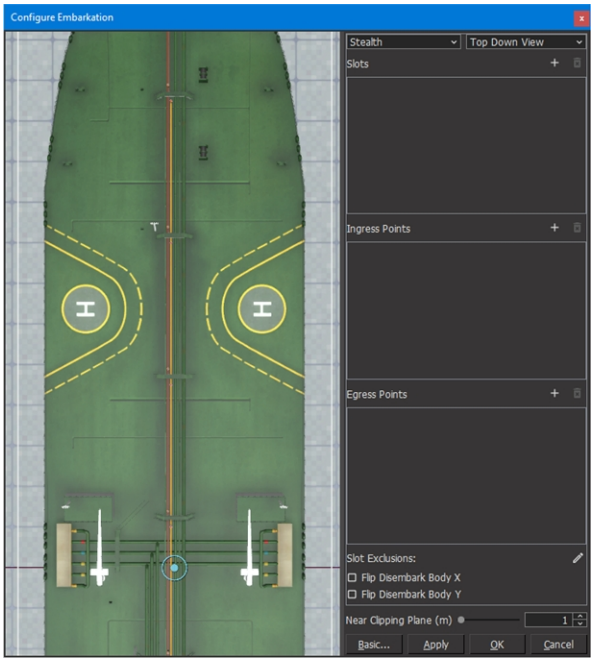
We want to configure specific slots for the super tanker. We cannot do that in the basic Configure Embarkation dialog box. The key parts of configuring a slot are specifying the object type and the location. Other parameters can provide more specific behaviors.

Add a Slot

➤ **To add a slot for the super tanker:**

1. In the list at the upper right of the dialog box, change the observer view from Plan View to Stealth. You now see a view of the super tanker model.
2. From the view list at the upper right, select Top Down View.
3. Zoom in so that you can see the forward half of the super tanker. In particular, you want to see the two helipads (Figure 616).

Figure 616. Stealth view



4. Click the Add button (+) above the Slots window. Slot 1 is added to the list.
5. Select Slot 1.

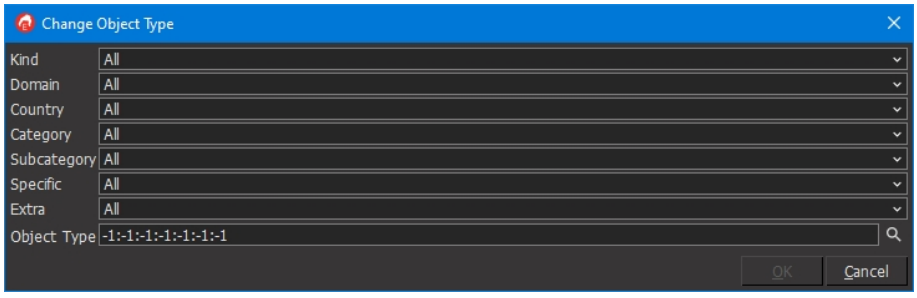
Specify the Object Type

We need to specify the object types of the entities that can use the slot. In this case, we just need to specify helicopters. However, helicopters have several object type enumerations. Helicopters can have category values of 20-25 (military), 50 (UAV), and 90-92 (civilian). You can specify multiple values in an enumeration using dashes for ranges and separating values with commas. For the helicopter, we want an enumeration that looks like this: 1:2:-1:20-25,50,90-92:-1:-1:-1. We will let the Simulation Object Editor build a basic object type. Then we will customize it.

➤ **To: specify the object type:**

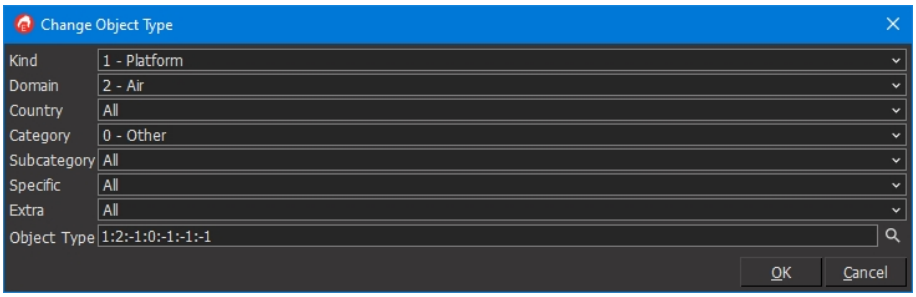
1. On the Object Type line, click the Select button (✎). The Change Object Type dialog box opens.

Figure 617. Change Object Type dialog box



2. In the Object Type dialog box, in the Kind list, select Platform (1).
3. In the Domain list, select Air (2).
4. In the Country list, select All (-1).
5. In the Category list, select Other (0).
6. For all the other lists, select All (-1). The results in an enumeration of 1:2:-1:0:-1:-1:-1 (Figure 618).

Figure 618. Completed base enumeration



7. Click OK to close the Change Object Type dialog box.

   The enumeration is listed in the Object Type box of the Configure Embarkation dialog box.
8. In the category part of the Object Type, replace 0 with 20-25,50,90-92. The completed object type is: 1:2:-1:20-25,50,90-92:-1:-1:-1 (Figure 619). This object type allows any type of helicopter to land in this slot.

Specify the Location of the Slot

We must specify the X, Y, and Z location of the slot relative to the origin of the super tanker. You will see a blue dot and yellow position text at the center of the ship. Please note that the coordinates specified in this example are good for the super tanker.

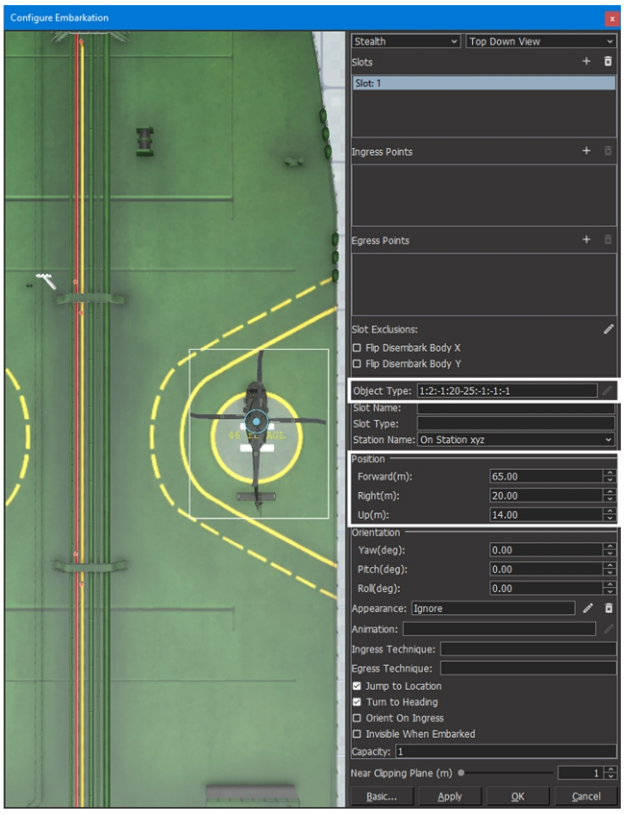
➤ **To specify the slot location:**

1. Make sure the Station Name is set to On Station xyz. We are going to specify the slot with an x,y,z offset.
2. Take a look at the location of the helipad to the right of the center line of the super tanker (starboard). Estimate the distance from the middle of the ship to the helipad.
3. Enter your estimated distance for the Forward (Y) value. A blue target circle displays on the center line of the ship.
4. Adjust the forward value to reach the proper distance (65 is a good value).
5. For the Right (X) coordinate, enter an estimated value for the distance to the right of center (starboard). (20 is a good value.) You should now see the target circle centered on the H of the helipad.
6. Enter a value for Up (Z). If you leave this blank, the helicopter will clamp to the deck of the tanker, but first it will sink to 0 meters, which doesn't look realistic. (14 is a good value for this coordinate.) An example aircraft displays on top of the

helipad.

7. Clear the Jump to Location check box. The dialog box should look like Figure 619.

Figure 619. Slot 1



8. Optionally, click Apply to save your changes.

Add a Second Slot

Add a second slot. Use the same object type and forward and up values. The Right position should be -20.

73.5.3 Add Ingress Points

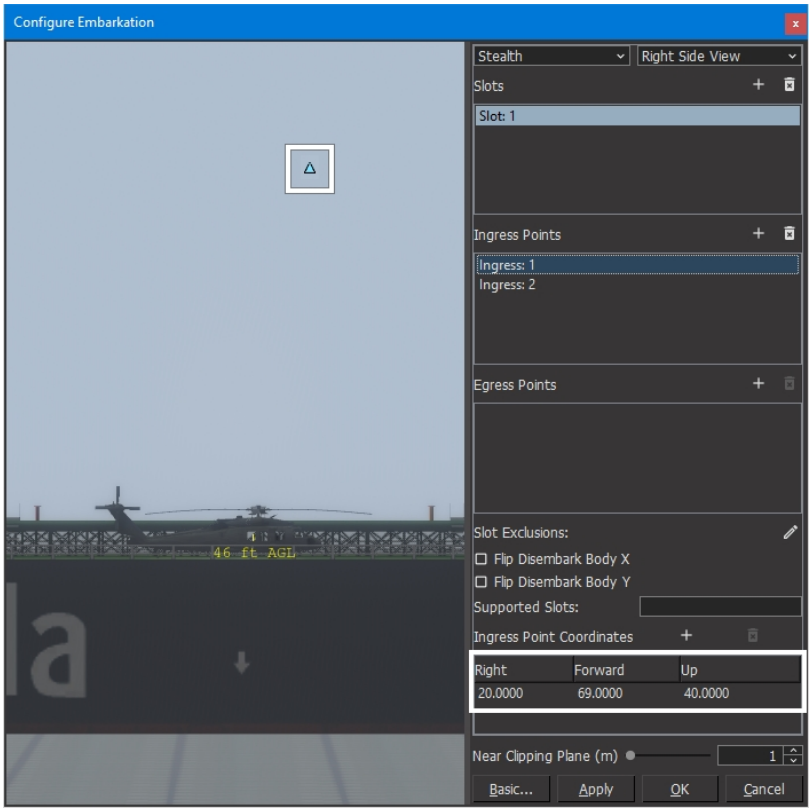
Ingress points define locations relative to the parent entity that an embarking entity moves to before it moves to its slot. While you do not have to define one, you should do so. If there are no ingress points, when you task an entity to embark, it teleports directly to a slot. It does not simulate movement towards the parent.

➤ **To add an ingress point:**

1. Click the Add button (+) above the Ingress Points window. A blue point displays at the center of the ship.
2. In the Ingress Points Coordinates table, double-click the Right value to make it editable.
3. Set it to 20 and then press Tab. The point moves to this location.

4. Set the Forward value to 69.
5. Set the Up value to 40. This will cause a helicopter to fly to a point above the helipad (the ingress point) and then land on the helipad. You can see the ingress point's position above the deck in one of the side views (Figure 620).
6. Add a second ingress point. Set Forward, Right, and Up to -20, 69, and 40.

Figure 620. Completed ingress points



7. Optionally, click Apply to save your changes.

73.5.4 Add Egress Points

Egress points define locations relative to the parent entity that a disembarking entity moves to after it leaves its slot. As with ingress points, egress points help manage realistic looking behavior for entities as they disembark. Without them, the entity simply stays where it is in its slot, but changes its state to disembarked.

➤ **To add an egress point:**

1. Click the Add button (+) above the Egress Points window. A red vertex marker is added to the ship.
2. Set the Right, Forward, and Up values to 20, 69, and 40. This will cause a helicopter to disembark and fly to a point above the helipad before beginning its next task. The vertex marker moves to show the location of the egress point.
3. Add a second egress point. Set Right, Forward, and Up to -20, 69, and 40.

73.5.5 Save the New Configuration

➤ **To save the configuration:**

1. In the Configure Embarkation dialog box, click OK.
2. In the main SOE window, choose **File > Save Simulation Model Set**.

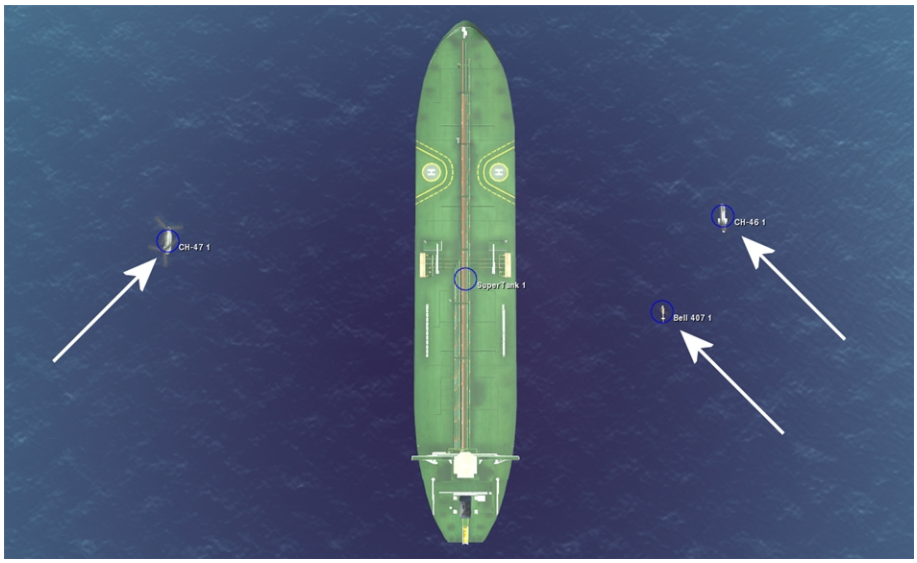
73.5.6 Test the Embarkation Configuration

Now you can test the new embarkation configuration for the super tanker.

➤ **To test the embarkation configuration:**

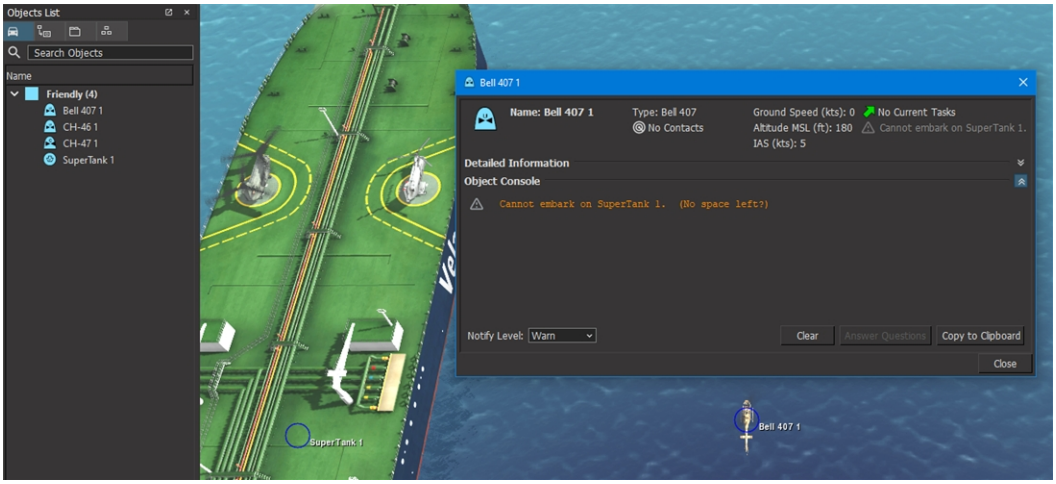
1. Start VR-Forces.
2. Create a new scenario using the entity-level SMS you created earlier and a terrain that has an ocean area, such as Ground_DB II or Ala Moana.
3. Place a Super Tanker in the water.
4. Add three helicopters. Put one on one side of the tanker and two on the other side. Use a variety of military and civilian helicopters. Set their altitude to about 180 - 300 feet.

Figure 621. Test scenario



5. Give each helicopter a plan to embark on the super tanker. Stagger the times at which they start their embark task so that they do not all start flying at once.
6. Run the scenario. Two helicopters should be able to embark in the two slots. The third should send a message to its console that it cannot complete its task because there are no slots available (Figure 622).

Figure 622. Helicopters embarked on Super Tanker





74. Adding DI-Guy Models to VR-Forces

This chapter explains how to take DI-Guy models created with the DI-Guy SDK and integrate them into VR-Forces.

74.1. Adding New DI-Guy Models to VR-Forces 1464
74.1.1 Copy DI-Guy Files to the Data Directory 1464
74.2. DI-Guy Configuration Files 1464
74.2.1 Adding a New DI-Guy Animation 1465
74.3. How VR-Forces Maps DI-Guy Models 1466
74.4. Add a New DI-Guy Entity in the SOE 1466

74.1. Adding New DI-Guy Models to VR-Forces

VR-Forces supports DI-Guy characters and includes many lifeform characters. However, MAK customers sometimes want to add their own DI-Guy characters. The general process for adding a new character is:

1. Create the new character using DI-Guy software and copy the required files to the appropriate *./SharedData/19/latest* subdirectory.
2. Add a model definition and element definition for the new character.
3. If the character has animations, add them to *character_animations_table.mtl*.
4. Create a new VR-Forces character in the Simulation Object Editor.

74.1.1 Copy DI-Guy Files to the Data Directory

VR-Forces stores files for DI-Guy characters in *./SharedData/19/latest/ModelData/Lifeforms/DIGuy* and its subdirectories. After you create a new character using DI-Guy software, you must copy the files for this character to these directories.

DI-Guy documentation recommends that all customer content be placed in *$(DIGUY)/custom*. To follow this convention, create a directory called *./SharedData/19/latest/ModelData/Lifeforms/DIGuy/custom* and put your files in the appropriate subdirectories under it.

VR-Forces can visualize custom DI-Guy characters based on DIS enumerations mapped to element definitions or by recognizing custom DI-Guy PDUs.

- If you want to map your custom characters to element definitions, then you must add new simulation models in the Simulation Model Editor. This will automatically create the complementary model and element definitions and object type mappings.
- If you want to visualize characters based on custom DI-Guy PDUs without creating element definitions or object type mappings, place the DI-Guy files in the *./SharedData/19/latest/ModelData/Lifeforms/DIGuy/custom* directory as described in the previous paragraph. No further configuration is needed.

74.2. DI-Guy Configuration Files

VR-Forces can specify the character, appearance, and animation values for DI-Guy models used in 3D visualization applications such as VR-Vantage and the VR-Forces GUI. A model's character determines the kinds of movement it can make. The appearance determines what it looks like. The animation specifies the movements that the character makes.

In the VR-Forces GUI, you can specify the animation with the DI-Guy Animation task. You can set the appearance with the DI-Guy Appearance set data request. For details about specifying DI-Guy characteristics in the GUI, see "32.13. DI-Guy Animation" on page 654, "40.25. DI-Guy Characteristics (Appearance, Head, Body Weight)" on page 876, and "40.26. DI-Guy Hand Item" on page 876.

You can specify the default character, appearance, and animation for a lifeform entity in the Simulation Object Editor. (For details, see "32.13. DI-Guy Animation" on page 654 and "70.1.10 Editing a Lifeform's Visual Model and Animation" on page 1383.) The DI-Guy mappings are stored in the entity's *.entity* file.

The DI-Guy animations available to lifeforms are controlled by *./appData/settings/vrfSim/character_animations_table.mtl*. This information is used to populate the lists in the Simulation Object Editor. You can edit this file to add additional DI-Guy characters to those already configured for VR-Forces. The syntax of entries is described in the file.

> ⚠ Any characters you add to *character_animations_table.mtl* must be valid DI-Guy characters. You cannot arbitrarily make up new types of characters. If you create new characters using a supported version of the DI-Guy SDK, MAK ONE applications will support these characters.

74.2.1 Adding a New DI-Guy Animation

This section explains how to add a new DI-Guy animation to VR-Forces. It assumes that you have the DI-Guy software and have created the new animation.

> ⓘ The new animation must be made with the same version of DI-Guy supported by VR-Forces. Please see the *VR-Forces Release Notes* for the currently supported version.

➤ **To add a new DI-Guy animation:**

1. Create the new animation in DI-Guy. You will create an animation and add it to a character's action table. Make sure that the name of the animation file matches the name of the action you added to the action table. For example, if the name of the animation is layEgg, but you call the action lay_egg, it will not work.

   > ⓘ There is a 1-1 relationship between characters and their action tables. You cannot use an animation based on a chicken character in a cow character's action table.

2. Add the *motion_name*.bdm file for the animation to *./data/Lifeforms/DIGuy/motions*.
3. Add the *action_table*.cfg file to *./data/Lifeforms/DIGuy/config/diguy*.
4. Add the new animation to *./appData/settings/vrfSim/character_animations_table.mtl*. This table determines whether an animation can be assigned to a character using the DI-Guy Animation task (**is-taskable-animation True**) or whether it is used based on the entity's behavior. For example, if a character is moving faster than a certain speed, it is given a running animation. An entry for our lay egg example would look like the following:

```
(chicken
  (display-name "Chicken")
  (animations
    (animation
```

```
        (animation-name "lay_egg")
        (name-to-be-displayed "Lay an egg")
        (is-taskable-animation  True)
        (animation-variant "normal")
      )
      (animation
        (animation-name "feeding")
        (name-to-be-displayed "Feed")
        (is-taskable-animation  True)
        (animation-variant "normal")
      )
...
```

74.3. How VR-Forces Maps DI-Guy Models

Ordinarily, VR-Forces chooses animations to play for a DI-Guy based on the lifeform's speed, position, and similar attributes. However, VR-Forces can specify the DI-Guy state and appearance with the Set DI-Guy Appearance set data request and the Set DI-Guy Animation task. Once you explicitly set the DI-Guy appearance or animation, VR-Forces no longer tries to choose animations and expects to be told how to render that lifeform for the remainder of the simulation.

74.4. Add a New DI-Guy Entity in the SOE

Add a new entity in the Simulation Object Editor, as described in "69.3. Creating a New Simulation Object" on page 1360.

When you specify the 3D model, the list of character names is drawn from the **display-name** attribute in the entries in *./appData/settings/vrfSim/character_animations_table.mtl*. Therefore, it must have been updated before you can specify the 3D model in the Simulation Object Editor.



75. The Object Parameter Database

This chapter provides a detailed description of the object parameter database. It is most useful to VR-Forces users who need to develop and configure new components.

75.1. The Object Parameter Database 1468
75.1.1 VR-Forces Classes and Object Parameters 1469
75.1.2 Object Types 1469
75.1.3 Parameter Types 1472
75.1.4 Local and Remote Subcomponents 1473
75.2. Component Descriptors and Parameters 1477
75.2.1 Common Elements of Component Descriptors 1478
75.2.2 Configuring the Priority of Components 1481
75.3. Platform Files 1481
75.4. System Definition Files 1482
75.4.1 Creating a System Definition File 1482
75.4.2 Configuring System Connections 1483
75.5. Adding Variable Bindings to Platform and System Files 1486
75.5.1 Adding a Variable Binding 1487
75.5.2 Making Variables Editable in the Simulation Object Editor 1488
75.5.3 Adding Platform Variable Bindings to the SOE 1488
75.5.4 Adding System Definition File Variables to the SOE 1490
75.6. State Properties and Parameter Properties 1490
75.6.1 Adding State Properties and Parameter Properties 1490
75.6.2 Getting and Setting State Properties Using Lua Scripts 1493
75.6.3 Displaying State Property Values in the Information Dialog Box 1493
75.6.4 An Example of Adding State Properties 1494
75.7. Configuring Units 1501
75.8. Configuring Tactical Graphics 1501
75.9. Pathname Interpretation in Simulation Model Set Files 1502

75.1. The Object Parameter Database

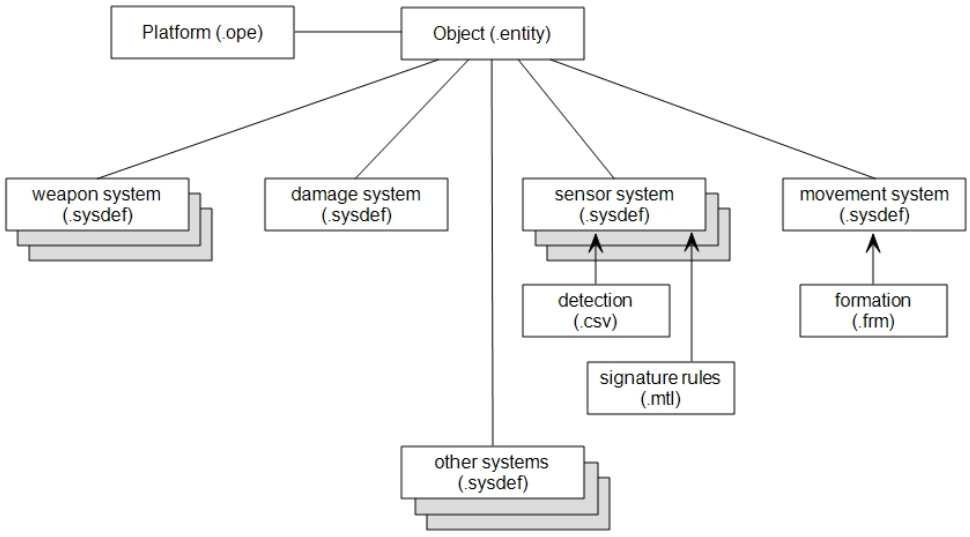
VR-Forces simulation objects and tactical graphics have many parameters. Parameter values are stored in files organized in simulation model sets (SMS) that VR-Forces reads when you load a scenario or create new objects. We refer to these files collectively as the object parameter database (OPD). The object parameter database specifies both the characteristics of a simulation object and starting state values. You can modify the behavior of a simulation object by changing the parameters for that type of simulation object. You can add new types of simulation objects and new behaviors to VR-Forces by adding new files to the object parameter database.

> ⚠ Before you make any changes to the object parameter database, make a backup copy of the file you are changing.

Figure 623 illustrates the organization of an entry in the OPD. The primary definition of a simulated object, such as an entity, a tactical graphic, or a cultural feature, is stored in a file with the extension *.entity*. The *.entity* file specifies the object type enumeration, marking text, organizational information, and other parameters. It references a platform file (extension *.ope*), that provides a set of generic components for the object type, such as ground platform, surface platform, or line object. Each *.entity* file also specifies the systems that allow it to interact with the simulated environment, such as sensors, weapons, and movement systems.

Systems are configured in system definition files. System definition files group related component and connection specifications into convenient packages. The components reference additional configuration files. The system definition files also make it possible to add systems to simulation objects in the Simulation Object Editor.

Figure 623. Object definition



75.1.1 VR-Forces Classes and Object Parameters

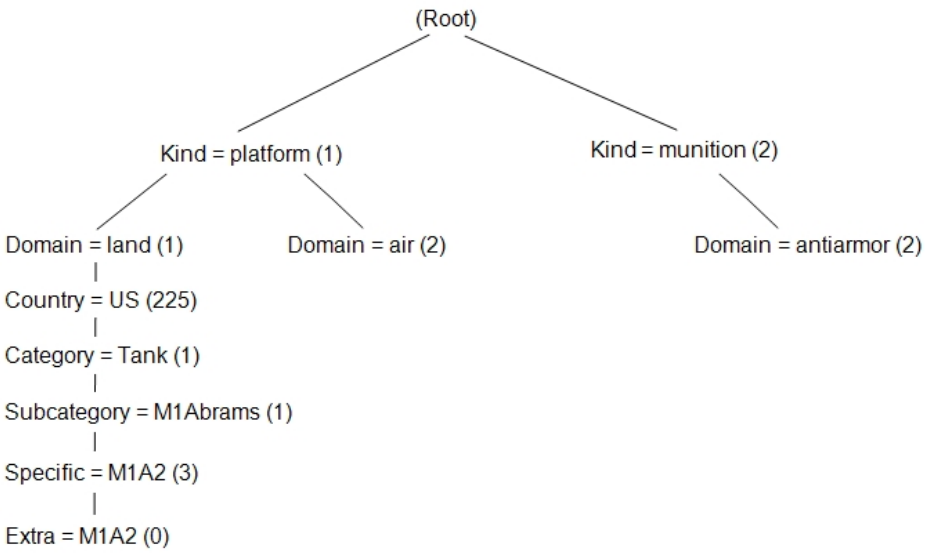
When you create any simulated object in VR-Forces, an instance of *DtVrfObject* is created in the simulation engine. Different types of simulation objects require different parameters for simulation and publishing, which are initialized by a subclass of *DtVrfObjectParameters*, and stored in a subclass of *DtVrfObjectBaseStateRepository*. Each *DtVrfObject* "has a" *DtVrfObjectParameters* and a *DtVrfObjectBaseStateRepository*. The type of parameters object to use for simulation object creation is specified in the simulation object's platform file by the **parameter-type** parameter. The state repository is specified in the **state-repository** parameter in the local-objects block. For example, when an entity that references the platform *Ground_Vehicle.ope*, which has a parameter type **ground-vehicle-param**, is created, a *DtVrfGroundVehicleParameters* object is created to read in the initial values from, and a *DtGroundVehicleStateRepository* is used to store runtime values.

75.1.2 Object Types

VR-Forces uses an eight-digit enumeration scheme for specifying objects. The enumerations are based on the seven-digit enumerations used by DIS and the RPR FOM. The additional field distinguishes between individual and unit objects.

This enumeration scheme describes a tree that goes from a more general specification to a more detailed one. Figure 624 illustrates a partial object type tree. To build an object type enumeration, start at the Kind and work your way down the tree, adding the number for each element. So, for example, if you follow the tree down to M1A2 you build the enumeration: 1 1 225 1 1 3 0. Since M1A2 is a leaf in the tree, the last field is 0. For more information about the object enumeration scheme, see the SISO Enumerations Document (*SISO Reference for Enumerations for Simulation Interoperability*).

Figure 624. Object type enumeration tree



The Object Super-Type

VR-Forces adds a field to the standard 7-digit entity enumeration so that we can manage objects that are not part of the DIS/RPR FOM enumeration scheme, such as units and tactical graphics. We enclose the seven-digit enumeration in parentheses and preface it with the object super-type (Figure 625). For example, the object enumeration for an M1A2, described in the previous paragraph, becomes the object type: 1 (1 1 225 1 1 3 0).

Figure 625. VR-Forces object-type parameter

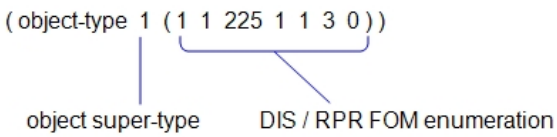


Table 68 lists the values VR-Forces can use for the object super-type. For more information, see *objTypeEnums.h* or "D.3. Object Type Enumerations" on page 1688.

Table 68. Object super-type field values

Object super-type	Meaning
0	Other.
1	Individual (platform entities, individual tactical graphics).
2	Unit.
3	Disaggregated unit (a unit composed of other simulation objects).
4	Aggregated unit (a unit that does not have subordinate simulation objects).

Published Object Types and Matching Object Types

Each simulation object in the OPD has two object types – the published object type and the matching object type.

The published object type is used as follows:

- When the GUI sends a create simulation object message to the sim engine, the sim engine uses the object type to determine which model to create.
- When the sim engine publishes a simulation object on the network it uses the published object type as the object enumeration.

Each field of a published object type must have a specific value.

The matching object type is used as follows:

- When the sim engine receives a request to create a simulation object but it cannot find an exact match for the object type that it is sent, it finds the best match among matching object types to determine which model to create. (For details about the

best match method, see "The Best Match Method" below.)

- When the sim engine discovers a simulation object on the network, it finds the best match among matching object types so that it can create local representative *DtVrf-Objects*.

The matching type can be the same as the published object type or it can have wildcards (-1).

For example, in VR-Forces, the published object type for an A-10 Thunderbolt is 1:2:225:2:4:1:0. If the matching type for the A-10 Thunderbolt is also 1:2:225:2:4:1:0 and VR-Forces discovers a remote simulation object with that same object type, it can match it to the A-10 Thunderbolt simulation model and can represent it properly in the simulation. However, if VR-Forces discovers a remote simulation object with the object type 1:2:225:2:4:1:1 (which is just a variation of an A-10), it will not be able to match the remote entity to an A-10 model because there is no A-10 match for that object type. (VR-Forces would end up matching the remote entity to a more generic fighter model, as explained in"The Best Match Method" below.) But if the A-10's matching object type is 1:2:225:2:4:1:-1, then VR-Forces will be able to match the remote entity to the A-10 model using the best match method.

Object types are specified in the *.entity* file:

```
<simObject objectType="1:1:2:225:2:4:1:0" matchType="1:1:2:225:2:4:1:-
1" platform="@(platforms-dir)/Fixed_Wing_Aircraft.ope">
```

The **objectType** attribute specifies the published object type; **matchType** specifies the matching object type.

The Best Match Method

As explained in "75.1.2 Object Types" on page 1469, object types describe a simulation object in increasing specificity as the object type fields move from left to right. Objects in the object parameter database are specified in a hierarchy in which entries at higher levels of the hierarchy have wildcard values (-1s) in object type fields that do not need to be specified at that level of the hierarchy. When VR-Forces looks up an object type, it begins at the root of the tree, working its way down until no better matches are found (the best match method.)

Suppose the object type hierarchy for fighter jets looks like this:

- -1:-1:-1:-1:-1:-1:-1:-1 – match any object.
- 1:1:2:-1:-1:-1:-1:-1 – match any fixed-wing entity.
- 1:1:2:225:2:-1:-1:-1 – match any U.S. attack fixed-wing.
- 1:1:1:2:225:2:4:1:-1 – match any A-10 Thunderbolt. (In VR-Forces, this is the A10 Thunderbolt matching object type.)
- 1:1:1:2:225:2:4:1:0 – match a specific A-10 Thunderbolt. (In VR-Forces, this is the A10 Thunderbolt published object type.)

As you replace each wildcard value (-1) with a specific enumeration, the scope of the object type narrows. Given this hierarchy, if VR-Forces discovers a simulation object with the object type 1:1:2:225:2:11:0:0, it tries to find the best match:

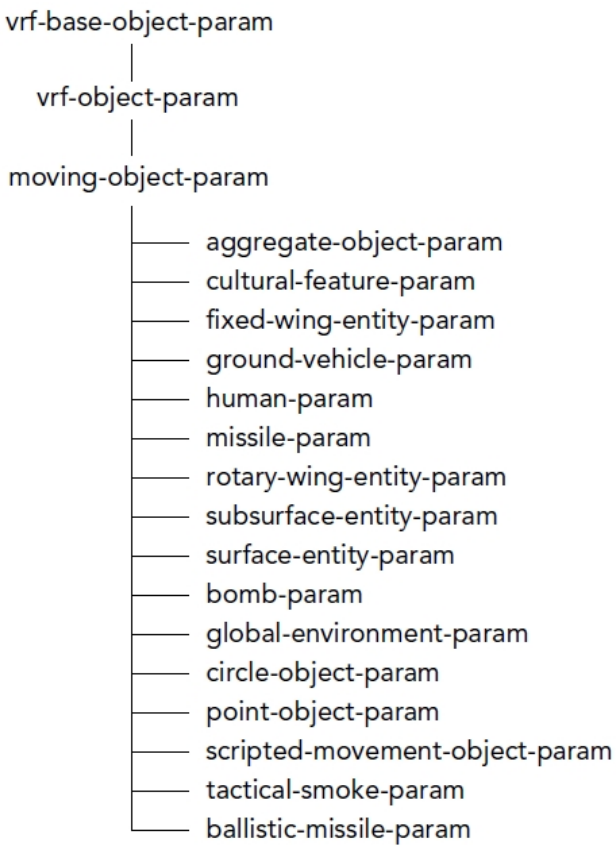
1. The discovered object type matches the top level - any object, because all fields are wild cards.
2. The discovered object type matches the fixed-wing object type, because the first three fields match and the rest are wild cards.
3. The discovered object type matches the U.S. attack fixed-wing object type.
4. The discovered object type does not match the A-10 Thunderbolt matching object type. Although the first five fields match, the sixth is different and is not wild carded.

The best match is the U.S. attack fixed-wing object type, so VR-Forces will use the generic U.S. attack fixed-wing model to represent the discovered simulation object.

75.1.3 Parameter Types

In addition to specifying an **object-type** for each object, you must specify a **parameter-type** for each object. The `parameter-type` determines the parameters and components that can be specified for an object. For example, a rotary-wing or fixed-wing entity can have a **nose-support** parameter, but a ground vehicle cannot. Figure 626 illustrates the object parameter type hierarchy.

Figure 626. Object parameter hierarchy



The `parameter-type` specifies the set of parameters appropriate to a simulation object, and the object state repository maintains state information for a simulation object, so the `parameter-type` and state repository must match. For more information, see "The State Repository Hierarchy" on page 1476.

> The **object-type** is specified in an object's *.entity* file. The parameter-type is specified in the platform (*.ope*) file.

75.1.4 Local and Remote Subcomponents

As explained in "3.3.2 Local Objects and Remote Objects" on page 120, each VR-Forces sim engine simulates objects itself (local objects), and maintains state information about remote objects. All of these objects are instantiated as *DtVrfObjects*. The *DtVrfObject* class has a "has a" relationship with the classes:

- *DtSimObjectStateRepository*
- *DtSimObjectNetInterface*
- *DtSimTaskManager*
- *DtSimComponentManager*
- *DtVrfObjectPlanManager*

These classes are referred to as subcomponents of the object. Table 69 maps the subcomponent classes to the parameters that specify them.

Table 69. Subcomponent classes and parameters

Class	Parameter
DtSimObjectStateRepository	state-repository
DtSimObjectNetInterface	net-interface
DtOrganizationManager	organization-manager
DtSimTaskManager	task-manager
DtSimComponentManager	component-manager
DtVrfObjectPlanManager	plan-manager

VR-Forces creates different subcomponents for an object depending on whether it is a local object or a remote object. The subcomponents are specified in the **local-objects** and `remote-objects` parameter blocks for that object. Table 70 lists the permissible entries for each parameter listed in Table 69.

Table 70. Permissible subcomponent parameters

Parameter	Permissible values
state-repository	• ballistic-missile-entity-state-repository • bomb-entity-state-repository • cultural-feature-state-repository • fixed-wing-entity-state-repository • ground-entity-state-repository • human-state-repository • missile-entity-state-repository • rotary-wing-entity-state-repository • surface-entity-state-repository • subsurface-entity-state-repository • vrf-aggregate-state-repository • vrf-moving-object-state-repository • vrf-object-base-state-repository • vrf-object-state-repository • vrf-overlay-object-state-repository • vrf-tactical-smoke-state-repository

Table 70. Permissible subcomponent parameters (continued)

Parameter	Permissible values
net-interface	• aggregate-local-entity-net-interface • engineering-object-local-net-interface • individual-local-entity-net-interface-with-properties • local-control-object-net-interface • local-control-object-net-interface-with-properties • local-control-object-net-interface • local-environmental-net-interface • local-tactical-smoke-net-interface • pseudo-aggregate-local-entity-net-interface-with-properties • engineering-object-remote-net-interface • vrf-aggregate-remote-entity-net-interface-with-properties • vrf-individual-remote-entity-net-interface • vrf-individual-remote-entity-net-interface-with-properties • vrf-remote-environmental-net-interface • vrf-remote-environmental-net-interface-with-properties • vrf-remote-tactical-smoke-net-interface
task-manager	• local-task-manager
component-manager	• component-manager
plan-manager	• vrfobject-plan-manager

Every object must have a state repository specified. Objects that should be published need a net interface. Other subcomponents are optional. However, for most simulation objects, you will want to specify a task manager, plan manager, and component manager so that you can control the simulation object and give it tasks.

The local and remote objects parameters are specified in the platform (*.ope*) files. Therefore they apply to all simulation objects that reference a particular platform. Use the existing entries in the object parameter database as guidance if you create a new platform or change an existing platform. A typical entry for local and remote objects is:

```
(local-objects
  (state-repository  "ground-entity-state-repository")
  (state-repository-min-tick-period  -1.000000)
  (state-repository-min-tick-period-variance  -1.000000)
```

```
    (net-interface  "individual-local-entity-net-interface-with
properties")
    (net-interface-min-tick-period  -1.000000)
    (net-interface-min-tick-period-variance  -1.000000)
    (task-manager  "local-task-manager")
    (component-manager  "component-manager")
    (plan-manager  "vrfobject-plan-manager")
    (nav-interface "active-bot")
  )
  (remote-objects
    (state-repository  "vrf-moving-object-state-repository")
    (state-repository-min-tick-period  -1.000000)
    (state-repository-min-tick-period-variance  -1.000000)
    (net-interface  "vrf-individual-remote-entity-net-interface-with
properties")
    (net-interface-min-tick-period  -1.000000)
    (net-interface-min-tick-period-variance  -1.000000)
    (task-manager  "")
    (component-manager  "")
    (plan-manager  "")
    (nav-interface "dynamic-obstacle")
  )
```

The State Repository Hierarchy

Each VR-Forces object must have a state repository to maintain essential state information about the object, such as its current location, orientation, appearance, and so on. Therefore, each object must specify a state repository and the state repository must match the **parameter-type** for the object. (The state repository and **parameter-type** are specified in the platform file.)

Table 71 lists parameter types and the state repository type required.

Table 71. State repository and parameter types

State Repository Type	Parameter Type
ballistic-missile-entity-state-repository	ballistic-missile-param
bomb-entity-state-repository	bomb-param
cultural-feature-state-repository	cultural-feature-param
fixed-wing-entity-state-repository	fixed-wing-entity-param
ground-entity-state-repository	ground-vehicle-param
human-state-repository	human-param
missile-entity-state-repository	missile-param
rotary-wing-entity-state-repository	rotary-wing-entity-param
surface-entity-state-repository	surface-entity-param
subsurface-entity-state-repository	subsurface-entity-param

Table 71. State repository and parameter types (continued)

State Repository Type	Parameter Type
vrf-aggregate-state-repository	aggregate-object-param
vrf-moving-object-state-repository	moving-object-param
vrf-overlay-object-state-repository vrf-object-state-repository	vrf-object-param
vrf-tactical-smoke-state-repository	tactical-smoke-param

The Net Interface Hierarchy

VR-Forces manages network relationships of local and remote simulation objects through the network interface classes. Choose the set of local and remote network interface types that most closely represent the type of object being simulated. For example, for an individual vehicle such as a tank, choose individual-local-entity-net-interface for the local network interface and vrf-individual-remote-entity for the remote network interface. See "Appendix D. Object Parameters" on page 1681 for tables that list the network interfaces used for the simulation objects provided with VR-Forces.

75.2. Component Descriptors and Parameters

The parameters for a simulation object include a list of all the sensor, controller, and actuator components it uses. (For details, see "13.5. VR-Forces Uses a Component Architecture" on page 366.) These components might be in the OPE file or in a system definition file. If there are no components of a certain type (for example, no sensors), you can omit the list. (In other words, you do not need a placeholder list for unused components.)

A component descriptor is a text block in a platform or system definition file that describes a component (see Example 1). A component descriptor must include the type of the descriptor and the type of the component that it is describing. A component descriptor can also contain parameters for the component. "75.2.1 Common Elements of Component Descriptors" on the next page describes the construction of a component descriptor.

Components can receive data from, and convey data to, other components through ports. These ports are specified in platform files and system definition files.

A component can create a port group, which groups a set of related ports together. For example, the automotive-control port group groups the throttle, steering, and brake ports. Use of port groups simplifies specification of connections because you just have to specify a port group instead of many individual ports. For a detailed discussion of ports and port groups, please see API documentation.

The port group also provides a mechanism for ensuring that only one data provider attempts to put values on those ports each simulation frame. For more information, see "75.2.2 Configuring the Priority of Components" on page 1481.

75.2.1 Common Elements of Component Descriptors

The common elements of component descriptors are:

- Name – The first element in a component descriptor specifies the component's name. This name can be whatever you want, but it must be unique within the simulation object or the system the component is defined in. It is used in the connections section of the simulation object's parameters. Other components may use this name to find the specified component. In Example 1, the name of the first component is move-along (line 3).
- `component-descriptor-type` – The component descriptor factory creates the type of component descriptor specified by this type. This lets the system know what types of additional data may be present. In Example 1, the component descriptor type for the move-along component is ground-move-to-descriptor (line 4).
- `component-type` – The component factory creates the type of component specified in this entry. In Example 1, the component type is ground-auto-move-along-controller (line 5).



The **component-descriptor-type** and **component-type** values are specified in code. You cannot make them up. See *compTypes.h* and *compDescTypes.h*.

- connections (optional) – The connections (Example 1, line 81) entry specifies how the components connect to one another (through ports, port groups, or both). Components attach to each other by connecting output and input ports to one another, or output port groups to input port groups. Since a component can have more than one port, port group, or both, the name of the port or port group to use is also specified. If a simulation object does not have any components that need to connect to one another, then you can omit this entry. Items in the connections entry can be in one of the following forms:
- Specify a port-to-port connection:

  `connect` *`component_name`*`:`*`output_port`* *`component_name`*`:`*`input_port`*

- Specify a port group-to-port group connection:

  `connect` *`component_name`*`:`*`output_port_group`*
  `component_name``:`*`input_port_group`*

  The connections in Example 1 use this format.

- Specify a port-to-port connection, in which one or both of the ports exists inside of a group:

  The input port belongs to an input port group:

  `connect` *`component_name`*`:`*`output_port`*
  `component_name``:`*`input_port_group`*`:`*`input_port`*

  The output port belongs to an output port group:

  `connect` *`component_name`*`:`*`output_port_group`*`:output_port`
  `component_name``:`*`input_port`*

Both input and output ports belong to groups:

```
connect component_name:output_port_group:output_port
  component_name:input_port_group:input_port
```

> The port and port group names are defined in code for each component. Please see class documentation for the names of port and port groups used by a given component.

Example 1 shows component descriptors for the Move Along and Patrol Between Points controllers and the Powertrain actuator for a ground vehicle (from the *ground-tracked.sysdef* file.) The Powertrain actuator simulates movement of a ground vehicle based on the throttle, brake, and steering inputs it receives from the automotive controllers. The example also shows how the output port groups for the controllers are connected to the input port group of the powertrain.

Example 1

```
(controllers
  ...
  (move-along
    (component-descriptor-type "ground-move-to-descriptor")
    (component-type "ground-auto-move-along-controller")
    (min-tick-period -1.000000)
    (min-tick-period-variance -1.000000)
    (periodic-tick-while-paused False)
    (process-state-repository-name "")
    (process-state-repository-type "")
    (is-enabled True)
    (is-enabled-for-clutter-entities True)
    (debug-detail False)
    (create-component True)
    (create-on-remote-object False)
    (parameter-data
    )
    (state-data
    )
    (state-and-parameter-data-prefix "")
    (script-ids )
    (script-variables )
    (near-distance 15.000000)
    (at-distance 1.000000)
    (approach-speed 2.000000)
  )
  ...
  (patrol-between-points
    (component-descriptor-type "ground-move-to-descriptor")
    (component-type "ground-auto-patrol-between-controller")
    (min-tick-period -1.000000)
```

```
    (min-tick-period-variance -1.000000)
    (periodic-tick-while-paused False)
    (process-state-repository-name "")
    (process-state-repository-type "")
    (is-enabled True)
    (is-enabled-for-clutter-entities True)
    (create-component True)
    (create-on-remote-object False)
    (parameter-data
    )
    (state-data
    )
    (state-and-parameter-data-prefix "")
    (script-ids )
    (script-variables )
    (near-distance 15.000000)
    (at-distance 1.000000)
    (approach-speed 2.000000)
  )
) ;; end controllers
(actuators
  (powertrain
    (component-descriptor-type "automotive-component-descriptor")
    (component-type "automotive-actuator")
    (min-tick-period -1.000000)
    (min-tick-period-variance -1.000000)
    (periodic-tick-while-paused False)
    (process-state-repository-name "")
    (process-state-repository-type "")
    (is-enabled True)
    (is-enabled-for-clutter-entities True)
    (debug-detail false)
    (create-component True)
    (create-on-remote-object False)
    (parameter-data
    )
    (state-data
    )
    (state-and-parameter-data-prefix "")
    (script-ids )
    (script-variables )
    (default-art-part-list )
    (check-terrain-collisions False)
    (use-high-fidelity-ground-clamping True)
    (fuel-type $fuel-type)
    (fuel-efficiency $fuel-efficiency)
  )
  ...
) ;; end actuators
```

```
81 (connections
82   ...
83   (connect move-along:automotive-control powertrain:automotive-
   control)
84   (connect patrol-between-points:automotive-control
   powertrain:automotive-control)
85   ...
86 ) ;; end connections
```

The connection in line 83 would read: the move-along component sends output to its automotive-control output port group. The powertrain receives the move-along controllers output through its automotive-control input port group.

75.2.2 Configuring the Priority of Components

Each simulation frame, VR-Forces ticks each simulation object's components in the order in which they are listed in the platform and system definition files. If a component believes it should be providing data, it takes control of the receiving component's ports or port group. Once a component acquires the ports, no other component can send data to that component during that frame. Therefore the order in which components are listed provides a natural priority scheme.

For example, among the ground movement controllers, the collision avoidance controller is placed on the controller list before any of the other movement controllers. If this controller needs to provide input to the ground automotive actuator to avoid a collision, it takes control of the automotive-control port group on the next tick and overrides the controller that previously owned the port group (such as Move To or Move Along).

75.3. Platform Files

Platform files (extension *.ope*) are the foundational configuration files for all simulated objects. They provide the basic description of each type of simulation object that you can create, such as a ground vehicle, surface vessel, or human. They also define types of tactical graphics. All platform files are part of an SMS. A platform file includes all of the components that every simulation object of that type needs to have, such as collision avoidance, radios, script controllers, and so on.

A platform file does not include systems that vary among specific examples of the platform, such as weapon systems, movement systems, and damage systems.

A platform file uses variables for the parameters that need to be different for specific examples of a platform. For example, all ground vehicles have a maximum speed, but tanks and civilian vehicles have a different value for this parameter. The values for these variables are set in the Simulation Object Editor.

You can create and edit platform files using a text editor. Platform files use MTL syntax. Most VR-Forces users will not need to create new platform files. You may need to add components or change parameters to be variables.

75.4. System Definition Files

The Simulation Object Editor allows VR-Forces users to easily add component systems, such as weapons, sensors, and movement, to simulation objects without needing to understand the components, configuration files, and connections required to make these systems work. These systems are defined in system definition files (*.sysdef*). System definition files are referenced in a simulation object's *.entity* file. They are read by the SOE and can be edited in it.

System definition files have two major sections – data and metadata. The data is a list of the components and connections required to implement the system for a simulation object.

The metadata explains to the Simulation Object Editor how to present the system to the user and how to integrate it into the entity model.

Weapon systems and damage systems have two system definition files, a template file (*.template_sysdef*), which specifies components shared by similar types of weapons or damage systems, and a weapon or platform-specific system definition file (*.sysdef*).

75.4.1 Creating a System Definition File

If you want to create a new system definition file, MAK recommends that you copy an existing file and use it as a starting point for adding the components that you want to configure into a system. You can use the SOE to do this or a text editor.

The format of a system definition file is:

```
(system-name
  (controllers
  )
  (actuators
  )
  (connections
  )
  (resources
  )
  (meta-data
    (system-name "Spot Report Generator")
    (system-description "Allows the entity to send spot reports
     through its radio when its sensors detect other entities
     in the scenario.")
    (allowed-state-repository-types "all")
    (system-categories "other")
    (meta-data-entry-list
    )
    (parameter-data-list
    )
  )
)
```

As you can see, system definition files use MTL syntax. (For information about MTL syntax, see "B.2. MAK Technologies Lisp (MTL) Files" on page 1664.) The system-name is an arbitrary name that uniquely identifies the system. The component sections list the various controllers, actuators, sensors, and resources that are needed to build the system. The connections section lists the connections among the components.

> ⚠ You can call the system whatever you want, but the name must not start with a number.

75.4.2 Configuring System Connections

Within each system definition file, you list the connections among the components in the file. (For general information about connections, see "75.2.1 Common Elements of Component Descriptors" on page 1478.) For systems that are self-contained, for example movement systems, there is no need to connect to components in other systems. However, some systems need to connect to components outside their system. For example, a visual sensor needs a list of object types to detect. This section explains how to make these out-of-system connections.

To understand how system definition files and *.entity* files are integrated, we will look at:

- A **componentSystem** element in a *.entity* file.
- The related system definition file.
- The connection section in the system definition file.

The M1A2, which uses the Ground_Vehicle platform, is configured with a visual sensor (in *M1A2_Abrams_MBT.entity*):

```
<simObject objectType="1:1:1:225:1:1:3:0" matchType="1:1:1:225:1:1:3:-
1" platform="platforms\Ground_Vehicle.ope">
...
<componentSystem systemName="sensor" platform="@(system-
dir)/sensors/visual-sensor.sysdef">
    <bodyPosition paramName="sensor-position" right="0.57" forward="-
1.14" up="2.1"/>
    <real paramName="max-range">4000</real>
</componentSystem>
```

The sensor component references the system definition file *visual-sensor.sysdef*. The parameters specified provide values for the variables in the system definition file. (For more information, see "75.5. Adding Variable Bindings to Platform and System Files" on page 1486.)

The system definition file has the following connections:

```
(visual-sensor-system
        ...
    (connections
        (connect system:object-types-to-detect visual-sensor:object-
types-to-detect)
```

```
        (connect visual-sensor:detected-objects system:detected-
objects)
        (connect system:sensor-offset visual-sensor:sensor-offset)
    )
    (resources )
    (meta-data
        (system-name  "Visual Sensor")
        (system-description  "Allows an entity to detect other objects
through visible light.")
        (allowed-state-repository-types "all")
        (system-categories "sensor")
...
)
```

The system definition file defines connections for its components. The first connection in the list (line 4) means that some external component (**system**) is providing input from its **object-types-to-detect** port to the visual sensor's **object-types-to-detect** port. The second connection specifies that the visual sensor's **detected-objects** port sends data to some external component's **detected-objects** port. When VR-Forces creates a simulation object, it needs to substitute an actual component for the generic "system" component specified in the system definition file. It does this by referring to an auto-connection block in the OPE file

The platform file has a set of auto-connection blocks like this (from *Ground_Vehicle.ope*):

```
(auto-connection
  (source-controller "target-selection")
  (source-port-name "object-types-to-detect")
  (destination-category "sensor")
  (destination-port-name "object-types-to-detect")
)
```

The auto-connection block in the OPE file is a template that VR-Forces uses to match systems of a specified category to the appropriate controllers. In this example, the **target-selection** controller sends data through its **object-types-to-detect** port to the **object-types-to-detect** port of any system that is in the sensor category. Since the visual sensor is in the sensor category, it substitutes the **target-selection** controller for the generic system in building its connections.

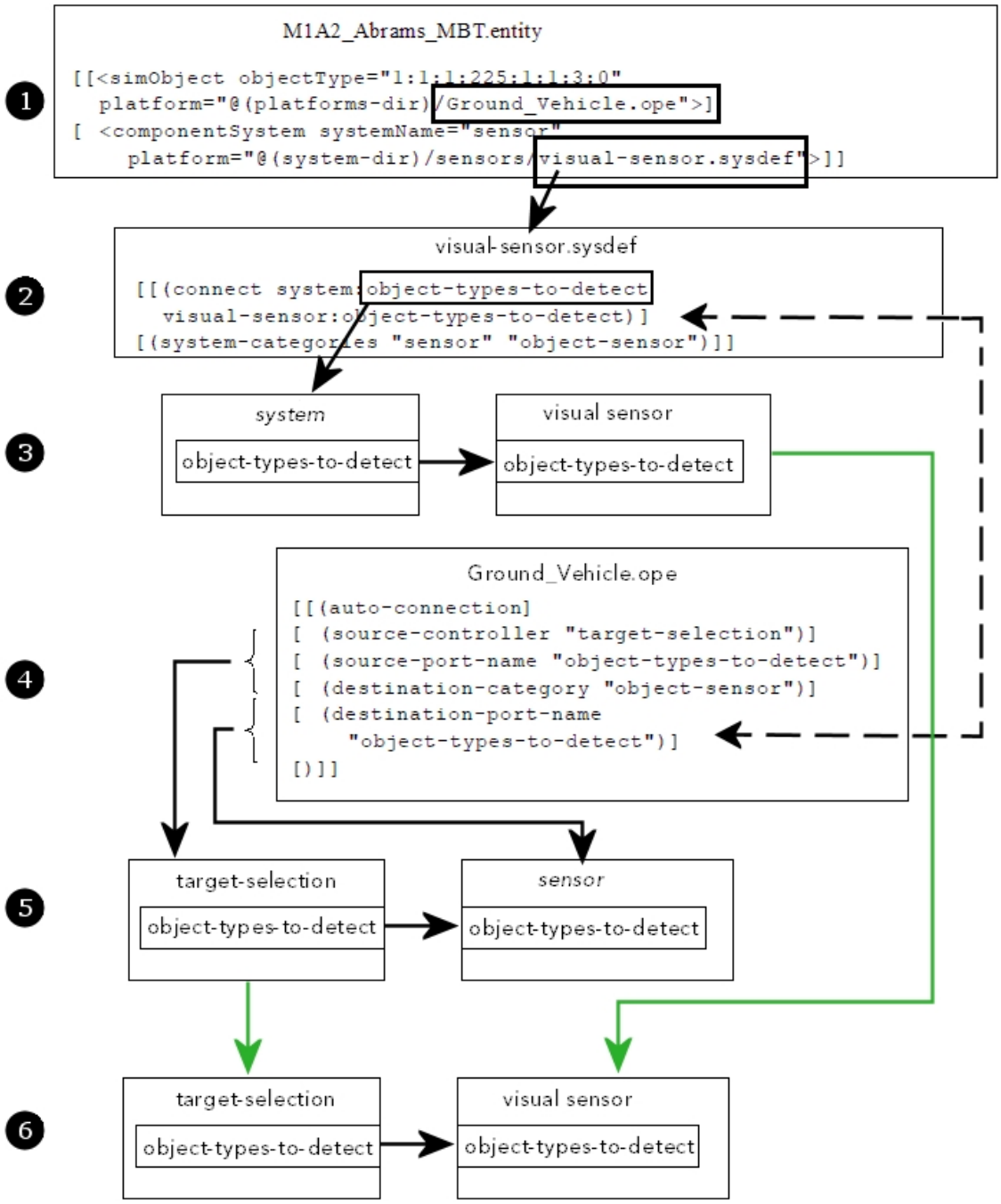
Figure 627 illustrates the relationships of the OPE file, *.entity* file, and system definition files referenced in this section and how connections are made.

The M1A2 entity, which uses the Ground_Vehicle platform, has a visual sensor (callout 1). The visual sensor has several connections. The one illustrated in the figure (2) takes input from an external source's **object-types-to-detect** port and sends it to the visual sensor's **object-types-to-detect** port (3). In this connection, the external source is represented by **system**. Before the sensor can actually work, this unspecified system must be determined.

When the entity is created, VR-Forces checks all the auto-connections in the OPE file (4). In this case, there is an auto-connection from the target-select controller's **object-types-to-detect** port to the **object-types-to-detect** port of a component whose destination category is **sensor** (5). To make this connection, the unspecified sensor must be determined.

VR-Forces matches the **destination-category** value "sensor" in the OPE file to the **system-categories** value "sensor" in the system definition file and the connections can now be completed (6).

Figure 627. External component connection



Systems that use templates must pass information between template files and individual system files. The "all" connection supports this exchange of information, as in this example:

```
(connections
```

```
    (connect system:all main-14_5mm-gun:all)
    (connect main-14_5mm-gun:all system:all)
  )
```

Most VR-Forces users add new systems by copying existing ones and changing their parameters. If you add new systems in this way, you do not need to worry about adding connections because the existing auto-connection system will handle them automatically. If developers add new types of components that require intersystem connections, they must add new auto-connection blocks to the appropriate OPE files for them to connect.

75.5. Adding Variable Bindings to Platform and System Files

Platform files and most system definition files are used by multiple simulation objects. However, the values of some parameters may need to vary for the different simulation objects that use them. This is handled by using variables in the platform and system definition files and specifying the values for these variables in the *.entity* files. VR-Forces uses variable bindings for the parameters that you are most likely to want to change from simulation object to simulation object. If there are other parameters that you want to vary, you can add your own variable bindings.

When you add a variable binding, you must:

1. Change the hard-coded value to a variable in the platform or system definition file.
2. Add the variable to the *.entity* file.
3. Update the Simulation Object Editor so that you can edit the value for the new variable.

This example shows a variable binding for fuel in a movement system. The variable is specified in the system definition file:

```
(resources
  (fuel
    (resource-type "real-resource")
    (amount  $fuel-amount)
    (full-amount  $fuel-amount)
  )
)
```

The *.entity* file for a simulation object that uses this movement system specifies a value for the variable:

```
<componentSystem systemName="movement"
      platform="@(system-dir)/movement/ground-tracked.sysdef">
  <real paramName="fuel-amount">465</real>
</componentSystem>
```

As you can see, the parameter defined as **fuel-amount** in the *.entity* file is referenced as `$fuel-amount` in the system definition file.

75.5.1 Adding a Variable Binding

The procedure for adding a variable binding is the same for platform files and system definition files.

➤ **To add a variable binding to a platform file or system definition file:**

1. In the system definition file, substitute a variable name, in the format $*variable-name* for the hard-coded value that you want to replace. For example, the *Ground_Vehicle.ope* file has a value **is-organized**:

   ```
   (groundParams
      ...
     (is-organized True)
      ...
   )
   ```

   To make this a variable, replace the value with a variable name:

   ```
   (groundParams
      ...
     (is-organized $entity-is-organized)
      ...
   )
   ```

   You can do this by hand in the OPE file.

2. Optionally, specify a default value for the variable. The default value is used if no value is specified for an entity type that uses this OPE file. The syntax for specifying a default value is:

   ```
   (default "value")
   ```

   In this example, to set the default value to True:

   ```
   (groundParams
      ...
     (is-organized $entity-is-organized (default "True"))
      ...
   )
   ```

3. In each *.entity* file that uses this platform file or system definition file, add an element to specify a value for the variable. The attribute must specify the type of the value, the name of the variable, and the value for the variable, for example.

   ```
   <simObject objectType="1:1:1:225:1:1:3:0"
         platform="platforms\Ground_Vehicle.ope">
     <bool paramName="entity-is-organized">True</bool>
   ...
   </simObject>
   ```

   Although you can add these entries by hand, it is highly recommended that you do so in the Simulation Object Editor by applying platform updates. When you apply platform updates, all changes to platform and system definition files are applied to all simulation objects that use them. For details, see "68.9. Applying New and Updated Variables" on page 1326.

> ⚠ If you do not add an attribute to an *.entity* file that references the modified platform or system definition file, a default value will be used, which may result in unexpected behavior.

75.5.2 Making Variables Editable in the Simulation Object Editor

When you add new variable bindings, you can make them editable in the Simulation Object Editor (SOE). Variables in platform files are edited in the General, Physical, and Movement tabs. They include variables such as speed, mass, and bounding volume. Variables in system definition files are edited in the Properties dialog boxes for the individual systems. The procedures for adding these two types of variables to the Simulation Object Editor are different.

75.5.3 Adding Platform Variable Bindings to the SOE

Variable bindings for platform files are edited in the General, Physical, and Movement tabs. The layout of these tabs is controlled by *.ui* files. The Simulation Object Editor has a *.ui* file for each platform. The files are in *./data/simulationModelSets/<model_set>/gui/ui*. You can edit *.ui* files in a text editor or in Qt Designer. Qt Designer is included with VR-Forces (*./bin64/designer.exe*).

> ⓘ You do not need a developer's license to use Qt Designer.

In "75.5. Adding Variable Bindings to Platform and System Files" on page 1486, we added a new variable binding to the *Ground_Platform.ope* file. The corresponding entry in *ground-vehicle-param.ui* file would be:

```
<item>
  <widget class="QCheckBox" name="entity_is_organized">
    <property name="text">
    <string>Is Organized</string>
    </property>
    </widget>
</item>
```

Notice that the variable name *entity-is-organized* in the OPE file is expressed as `entity_is_organized` in the *.ui* file. You must use underscores in the *.ui* file for the variable name.

If you are adding an attribute to the Simulation Object Editor and editing the .ui file by hand, the easiest approach is to copy an existing item, paste it where you want, and edit it for the new attribute. Many of these attributes have the **display-units** and **source-units** properties. For attributes that are expressed in units, such as meters or kilograms, these properties specify the unit. The unit value is expressed in the attribute label, for example (m) or (kg). If your new attribute is not expressed in units, leave these fields blank.

UI File Details

Whether you edit a *.ui* file by hand or using Qt Designer, you need to understand how data entry elements are defined. This varies depending on the type of widget used, such as text box, check box, option button, or list.

It is beyond the scope of this manual to explain Qt Designer and GUI layout. However, at the most basic level, the controls in the GUI are specified in item elements. Each item has a widget that has a **class** and an **objectName**. (The XML attribute is **name**. In Qt Designer it is **objectName**.)

- The **objectName** for an item is the platform variable name with dashes changed to underscores. For example, the **objectName** for the `short-name` variable is `short_name`.
- If an item has a label, the **objectName** must be `<variable_name>_label`. For example, the **objectName** for the Short Name label is `short_name_label`.

Each data entry widget creates a data entry field that is appropriate for the variable type (integer, real, string). Other elements (systems, vector entry) create more complex controls.

Depending on the type of information displayed, you can assign additional properties to each data entry field. These properties can be:

- element_type. String property that can be assigned a particular reader/writer (variable) type (for example, DtRwReal) or any other internal type that has been registered for element creations. These types are:
- single_system_selector
- multi_system_selector
- sensor_signatures
- aggregate_composition
- aggregate_formation_editor
- push_button
- scripted-movement-system-selector.
- enable_if_checked. String property that defines a check box element. If the check box is selected, the variable is enabled. If clear, it is disabled.
- disable_if_checked. String property that will define a check box element. If the check box is selected, the variable is disabled. If clear, it is enabled.
- disable_if_multiple_selection. Boolean property. If selected, when the user selects multiple simulation objects, this item cannot be edited.
- hide_by_default. Boolean element. If selected, this item is hidden by default and is visible only if the Advanced button is clicked.
- signal_slot. String element. If defined for a push button, a particular slot is called when that signal is sent. The format is:

  ```
  <signal>;<referencedControl>;<slot>released();bounding_volume;
  ```

```
onCalculateSupportPointsChanged()
```

The referenced control is the name of an existing control in the *.ui* file.

75.5.4 Adding System Definition File Variables to the SOE

When you add a system to a simulation object in the Simulation Object Editor, you can edit its properties. The Properties dialog box is not configured in a *.ui* file. It is generated based on the metadata section of the system definition file. Suppose that you want to change the rounds per minute parameter in *120mm-gun.sysdef* to a variable binding. You would create the variable binding as described in "75.5. Adding Variable Bindings to Platform and System Files" on page 1486 and also add details to the metadata as shown in this example:

```
(weapon-120mm-gun
  ...
  (rounds-per-minute $rounds-fired-per-minute)
  ...
  (meta-data
    (parameter-data-list
      (bool-parameter-data
        (parameter-name  "rounds-fired-per-minute")
        (variable-type  "DtRwInt")
        (display-label  "Rounds Fired Per Minute")
        (display-units  "")
        (source-units  "")
        (default-value  True)
      )
...
```

75.6. State Properties and Parameter Properties

All simulation objects and tactical graphics have properties, such as ordered speed, mass, and force, that are stored in their state repository. You can add the following types of properties to these objects without using the VR-Forces Toolkit:

- State properties. A property that can store state information locally or can be published on the network. You can access the property using Lua scripts or the C++ API and you can change the value.
- Parameter properties. A state property whose value is static and cannot be changed at runtime.

The VR-Forces GUI and simulation components do not know anything about these state properties unless you write code to integrate them.

75.6.1 Adding State Properties and Parameter Properties

You define state and parameter properties for simulation objects that you want to have them. State properties are defined in the `state-data` section of the OPE files and parameter properties in the `parameter-data` section. You can also add state properties to an entity based on what sim components are added to the entity.

When you define a sim component in a sysdef, you can add a `state-data` or `parameter-data` section to the sim component description. When the entity creates that sim component, it automatically adds any defined state properties to the list of properties defined in its OPE.

The basic syntax for defining a state property is:

```
(type property_name default [publish record_type])
```

where:

- *type* is a data type. See Table 72 for a list of supported data types.
- *property_name* is the name of the property, such as NumberOfWheels.
- *default* is a default value.
- `publish` is an optional parameter that specifies that the state property should be published.
- *record_type* is an optional parameter to the publish flag. For details, please see "Publishing User-Created State Properties" in the *VR-Forces Developers Guide*.

This example specifies two state properties:

```
(state-data
  (DtRwInt TargetsMissed 0)
  (DtRwInt NumberOfWheels 4 publish)
)
```

Both properties are of type DtRwInt. The TargetsMissed property is initialized with the value 0. It is not published. The NumberOfWheels property has a default value of 4. It is published to the network.

You can bind state properties to variables. This allows you to specify different values for different simulation objects. For a complete discussion of binding properties to variables, see "75.5. Adding Variable Bindings to Platform and System Files" on page 1486. The format for specifying a state property and binding it to a variable is:

```
(type property_name variable [(default default_value) publish record_type])
```

where:

- *type* is a data type. See Table 72 for a list of supported data types.
- *property_name* is the name of the property, such as NumberOfWheels.
- *variable* is a variable name that you use to add the property to platform and system files.
- `(default` *default_value*`)` specifies a default value for a variable. (Optional.)
- `publish` specifies that the state property should be published. (Optional.)
- *record_type* is a parameter to the publish flag. For details, please see "Publishing User-Created State Properties" in *VR-Forces Developers Guide*. (Optional.)

This example adds a state property with a variable binding:

```
(state-data
  (DtRwInt NumberOfWheels $number-of-wheels (default 4) publish)
)
```

You specify parameter properties in the `parameter-data` section of the OPE file. The syntax is the same.

Table 72 identifies the data types supported for state properties. The Data Type column indicates the name of the data type used in the OPE file definitions. (The data types used by C++ are slightly different.) The Description column describes the format of this type. The HLA Representation/Encoding column indicates what type should be used in an HLA 1516 FOM to match the specification in the OPE. For information about HLA and state properties, please see "Publishing User-Created State Properties" in *VR-Forces Developers Guide*.

Table 72. Data types supported by state and parameter properties

Data Type	Description	HLA Representation/Encoding
DtRwInt, DtRwInt32	A 32-bit integer	RPRunsignedInteger32BE, HLAinteger32BE
DtRwInt16	A 16-bit integer	RPRunsignedInteger16BE, HLAinteger16BE
DtRwInt8	An 8-bit integer	RPRunsignedInteger8BE, HLAoctet
DtRwReal, DtRwReal64	A 64-bit float	HLAfloat64BE
DtRwReal32	A 32-bit float	HLAfloat32BE
DtRwBoolean	A Boolean	RPRboolean
DtRwString	An ASCII string	RPRnullTerminatedArray of HLAASCIIchar
C++: DtRwPropertiesFixedArray OPE: DtRwFixedArray	A fixed sized list of items	HLAfixedArray
C++: DtRwPropertySizedString OPE: DtRwSizedString	An ASCII string (alternate encoding for DtRwString)	HLAASCIIstring
C++: DtRwPropertiesStructure OPE: DtRwStructure	A fixed structure with multiple named fields	HLAfixedRecord with matching fields.
C++: DtRwPropertiesList OPE: DtRwList	A variable sized list of items	HLAvariableArray
C++: DtRwPropertiesMap OPE: DtRwMap	A variable sized map of items	HLAvariableArray of HLAfixedRecords, where the first field in each record is the key.

75.6.2 Getting and Setting State Properties Using Lua Scripts

You can set and get the values for state properties using scripted tasks and sets or through C++ code. For details about accessing them using C++, please see the *VR-Forces Developers Guide*. This section describes how to access state properties using Lua.

To get the value of a state property from Lua, use the getStateProperty() function of a SimObject. Use getParameterProperty() to retrieve parameter properties.

```
local hits = this:getStateProperty("TargetsHit")
```

To set the value of a state property from Lua, use the setStateProperty() function.

```
this:setStateProperty("TargetsHit", 2)
```

The setStateProperty() function has an optional third argument that specifies that the property should be created if it does not already exist. If not set, this argument defaults to false. When a state property is added this way, VR-Forces must try to determine what the type of the property is, and it might not make the choice you intended, especially for properties with complex types. Therefore, if you are defining a complex property we recommend that you add it to an OPE file. Properties added using this argument cannot be published to the network.

```
this:setStateProperty("SomeNewProperty", 1.2, true)
```

75.6.3 Displaying State Property Values in the Information Dialog Box

You can add user-created state properties to the State Data page in the Information dialog box.

➤ **To add state properties to the State Data page:**

1. Open *./appData/settings/vrfGui/default_guiConfiguration.gui* in a text editor.
2. Find the DtEntityInformationPage `pageitem` element.
3. Add an `informationitem` element for each state property you want displayed, in the format:

   `<informationitem informationitemname="DtStatePropertyPageItem:`*`MyStateProp`*`:`*`Display Name`*`"/>`

   where:

   — *`MyStateProp`* is the name of the state property as specified in the OPE file.

   — *`Display Name`* is the text that you want to display for the property in the Information dialog box.

75.6.4 An Example of Adding State Properties

This section is an example of how you can add state properties, configure them in the Simulation Object Editor, set them with a scripted set, and use them in a scripted task.

For this example, let's assume a scenario in which ground vehicles need to travel from one location to another. There are several routes they can follow. One route can only be traveled by smaller wheeled vehicles. A second can be traveled by large wheeled vehicles, defined as any vehicles with more than four wheels. A third can be traveled by tracked vehicles.

To make this all work, we need to know if a vehicle has wheels, and if so, how many. We will start by adding state properties to hold this information.

Add the State Properties

We want to add two state properties. `hasWheels` indicates that a vehicle has wheels. `numWheels` holds the number of wheels. We will add these properties to *./data/simulationModelSets/sms_name/vrfSim/platforms/Ground_Vehicle.ope* as variables so that we can add them to the Simulation Object Editor. `numWheels` has a default value of 4.

```
(state-data
  (DtRwBoolean hasWheels $has-wheels publish)
  (DtRwInt numWheels $number-of-wheels (default "4") publish)
  ...
)
```

Add the State Properties to the Simulation Object Editor

You do not have to make state properties settable in the Simulation Object Editor, but if you want to use them for a lot of simulation objects, it is convenient to be able to set them when you add new simulation objects or update existing ones.

Ground vehicle variables are configured in *./data/simulationModelSets/sms_name/gui/ui/ground-vehicle-param.ui*. Each variable is configured as an `item` element. We want to add the state properties so that we set `hasWheels` with a check box. If the check box is enabled, the `numWheels` field becomes enabled and we can set that.

If you take a look at the variables available to a ground vehicle in the Simulation Object Editor, you will see that many of them have check boxes and an associated value. You will also see that the Configure button is disabled unless Can Be Embarked On is selected. Using the existing item elements as a guide, we add the following code after Maximum Pivot Speed, so that it is listed after Maximum Pivot Speed widgets:

```
<item>
  <layout class="QHBoxLayout" name="horizontalLayout_66">
      <item>
          <widget class="QCheckBox" name="has_wheels">
              <property name="text">
                  <string>Has Wheels</string>
              </property>
```

```
                <property name="element_type" stdset="0">
                    <string notr="true">DtRwBoolean</string>
                </property>
            </widget>
        </item>
        <item>
            <widget class="QLineEdit" name="number_of_wheels">
              <property name="enable_if_checked" stdset="0">
                    <string notr="true">has_wheels</string>
                </property>
                <property name="source-units" stdset="0">
                    <string notr="true"></string>
                </property>
                <property name="display-units" stdset="0">
                    <string notr="true"></string>
                </property>
            </widget>
        </item>
    </layout>
</item>
```

We set the name for the `layout` element to a unique name (horizontalLayout_66).

The QCheckBox widget has the name `has_wheels`. This is the variable for the `hasWheels` property (`$has-wheels`), with the hyphen replaced with an underscore. (This convention is described in "75.5.2 Making Variables Editable in the Simulation Object Editor" on page 1488 and "75.5.3 Adding Platform Variable Bindings to the SOE" on page 1488.) The label for the check box is Has Wheels and it is of type DtRwBoolean, which matches the data type we set in the OPE file.

The QLineEdit widget (the data entry field) has the name `number_of_wheels` to reference the `$number-of-wheels` variable. The **enable_if_checked** property is set to `has_wheels`. This means if `has_wheels` is checked, `number_of_wheels` is enabled. We left the other properties blank since we do not express the number of wheels in any unit of measure.

> You can also edit *ground-vehicle-param.ui* in Qt Designer, which is included with VR-Forces (*./bin64/designer.exe*). When you add a widget for a variable, the objectName property should be set to the variable name, for example, `number_of_wheels`.

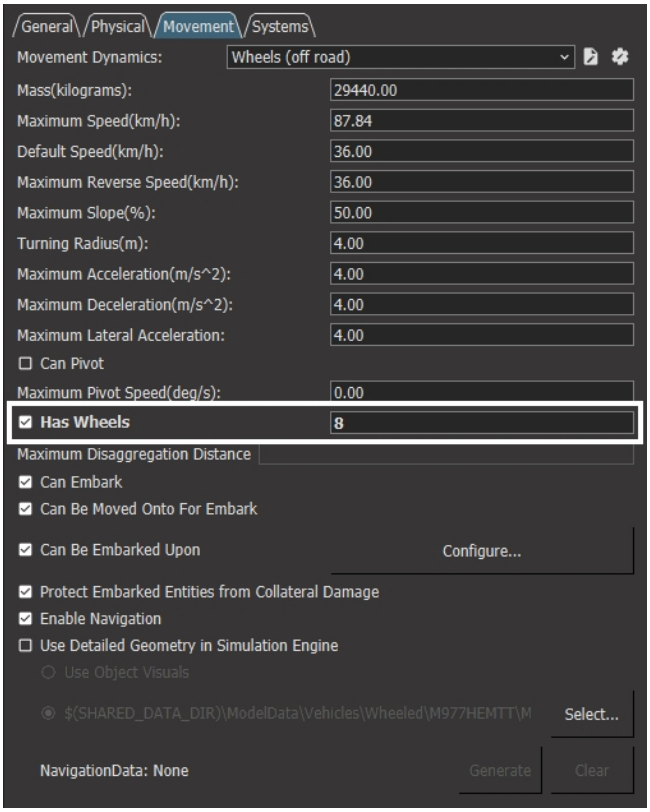
Update the Platform in the Simulation Object Editor

Now we need to update the Simulation Object Editor to use the updated *ground-vehicle-param.ui* file.

1. Start the Simulation Object Editor.
2. Load an entity-level SMS file that you can edit. If you do not have one, see "68.7. Creating New Simulation Model Sets" on page 1323 for instructions.
3. Choose **File > Regenerate SMS Files**. An information prompt opens.

4. Click OK. The Simulation Object Editor is updated so that all ground vehicles now have the Has Wheels check box and the number of wheels data entry field (Figure 628).

Figure 628. Has Wheels check box and data entry field



5. You can now configure vehicles to have wheels and set the number of wheels.

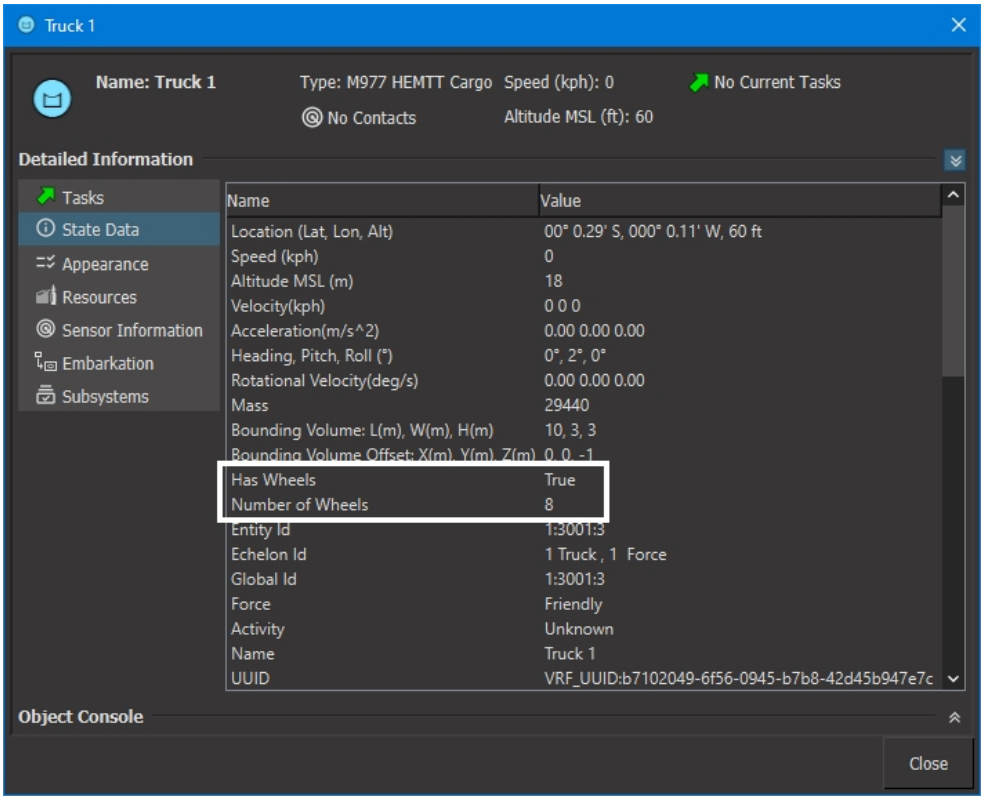
Add the State Properties to the Information Dialog Box

If you want to see the values for the state properties in the Information dialog box, add the following lines to the DtEntityInformationPage `pageitem` element in *./appData/settings/vrfGui/default_guiConfiguration.gui*:

```
  <informationitem
informationitemname="DtStatePropertyPageItem:hasWheels:Has Wheels"/>
  <informationitem
informationitemname="DtStatePropertyPageItem:numWheels:Number of
Wheels"/>
```

Figure 629 shows the Information dialog box for an entity that is set to have wheels.

Figure 629. Information dialog box with user state properties added

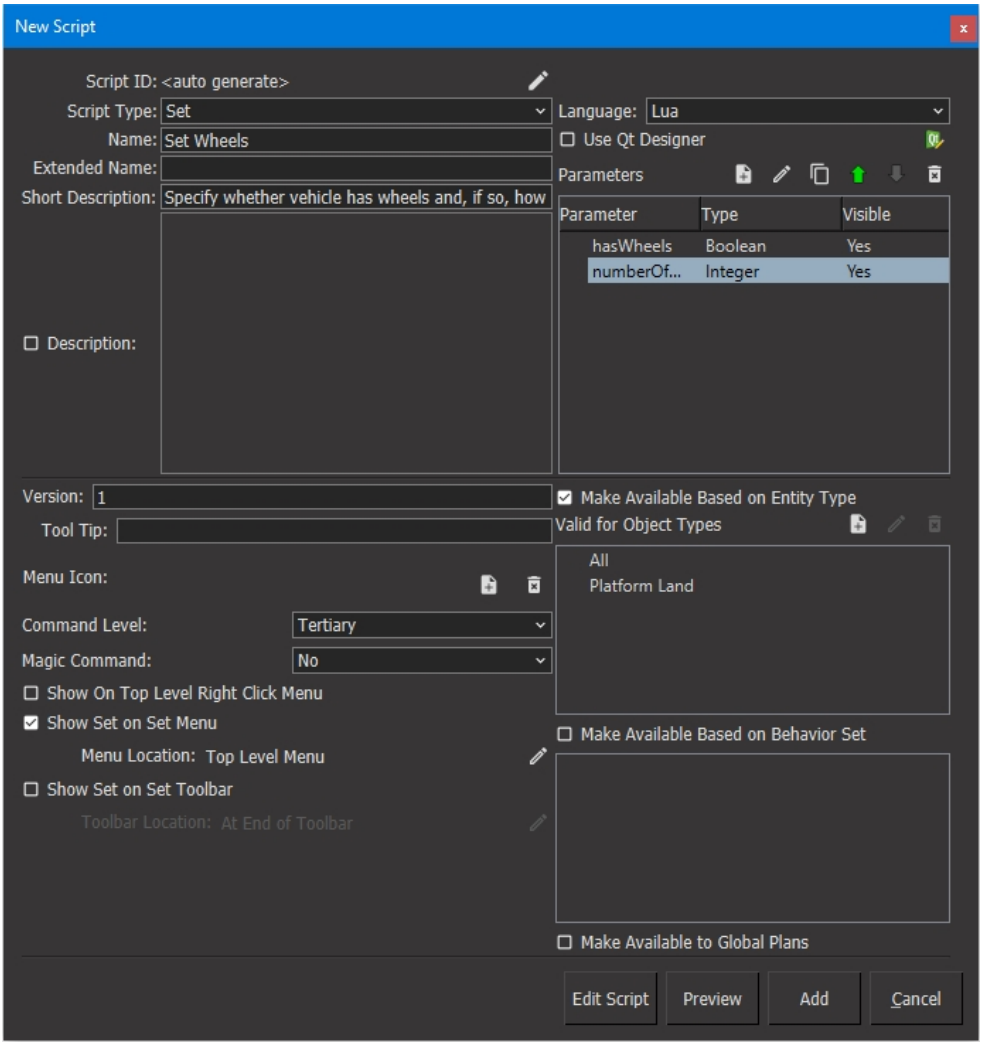


Write a Scripted Set to Set the Number of Wheels

You might not want to configure every ground vehicle in the Simulation Object Editor with its wheel status. Maybe you just want to be able to set this in the GUI. To do that you can write a scripted set data request.

Figure 630 shows the setup for the Set Wheels scripted set. It has two parameters that match the state properties we added. It is valid for land entities and is on the Set menu.

Figure 630. Set Wheels metadata



The Lua script for Set Wheels is as follows. It starts out by getting the current values for the two state properties and printing them to the object console. Then it takes in the input from the Set Wheels dialog box and sets the state properties to the new values. Finally, it gets the state properties and prints them to the console to confirm that it worked. We have left out the generic comments and the unused functions and commented the code.

```
require "vrfutil"
function init()
  vrf:setTickWhilePaused(true)
-- get the current values for the state properties
  local haswheels = this:getStateProperty("haswheels")
  local numwheels = this:getStateProperty("numwheels")
-- if the entity has wheels, print out that it has them and how many
-- this tests that we are getting the state properties correctly
  if haswheels then
      print("This vehicle has wheels.")
      print("Number of wheels is: ", numwheels)
```

```
    else
  -- if no wheels print this instead
        print("This vehicle does not have wheels.")
    end
    finished = 0
    vrf:setTickPeriod(0.5)
  end
  function tick()
  -- this happens after we complete the Set Wheels dialog box and
click OK
  -- get the haswheels value from the dbx and set the state property
    haswheels = setParameters.hasWheels
    this:setStateProperty("hasWheels", haswheels)
  -- get the number of wheels from the dbx and set the state property
    wheels = setParameters.numberOfWheels
    this:setStateProperty("numWheels", wheels)

  -- test it again. get the property values and print out the results.
    newhaswheels = this:getStateProperty("hasWheels")
    newnumwheels = this:getStateProperty("numWheels")

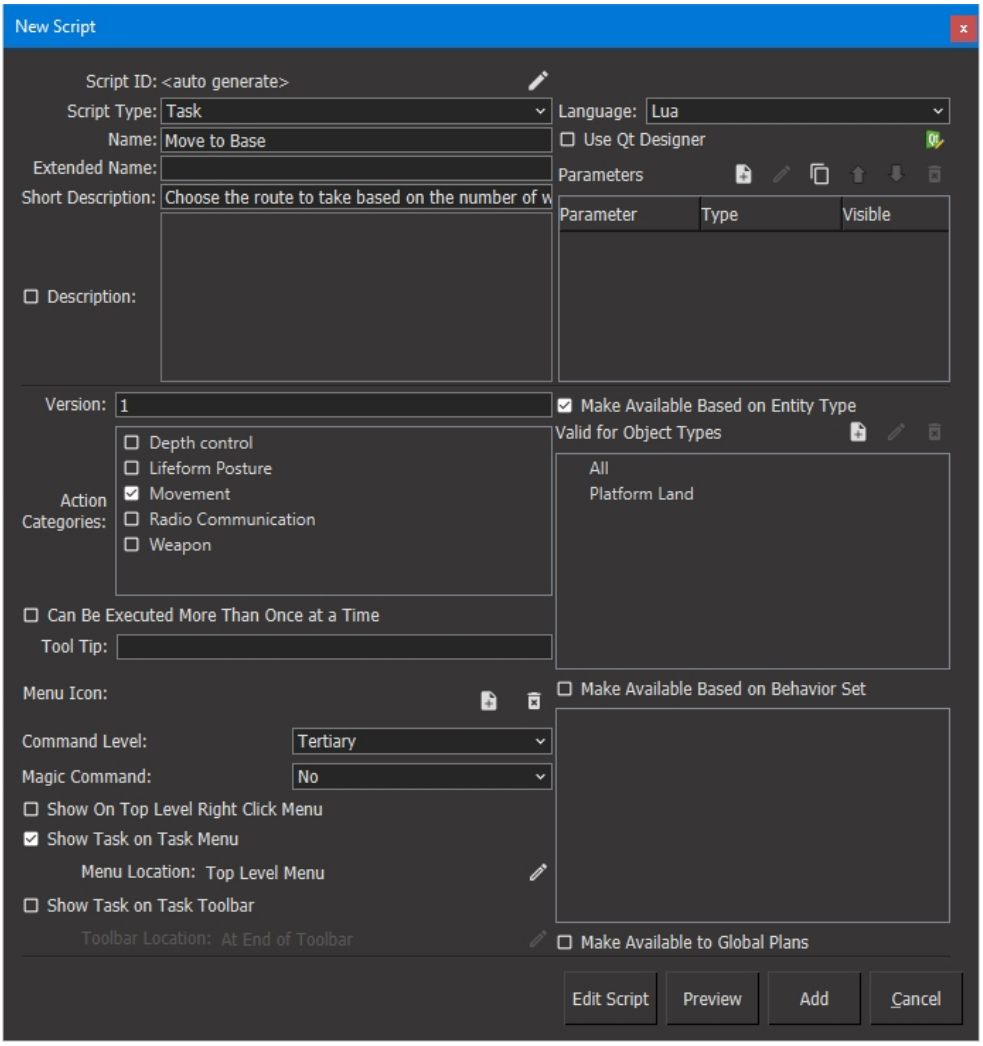
    if newhaswheels then
        print("This vehicle has wheels.")
        print("Number of wheels is: ", newnumwheels)
    else
        print("This vehicle does not have wheels.")
    end
    finished = 1

    if finished == 1 then
        vrf:endSet()
    end
  end
```

Write a Task that Uses the State Properties

Now we will write a script that uses the state properties. Remember that the premise of the exercise was based on the need for different vehicles to travel on different routes to reach a destination. In the scenario, we create three routes – Tracked, Heavy, and Light, that lead to the destination (Base). Our task, Move to Base, gets the `hasWheels` and `numWheels` state properties and based on what it finds gives a Move Along Route task to the entity using the appropriate route. There is no input to the task, so the New Scripted Task dialog box just sets up the name of the task and puts it on the menu (Figure 631).

Figure 631. Move to Base metadata



Here is the Lua script:

```
require "vrfutil"
function init()
  moveAlongRouteSubtaskID = -1

-- get the properties and print to the console for test purposes
  haswheels = this:getStateProperty("haswheels")
  numwheels = this:getStateProperty("numwheels")
  if haswheels then
      print("This vehicle has wheels.")
      print("Number of wheels is: ", numwheels)
  else
      print("This vehicle does not have wheels.")
  end
  vrf:setTickPeriod(0.5)
end
```

```
  function tick()
    if moveAlongRouteSubtaskID == -1 then
  -- see if the vehicle has wheels. If yes, give proper move along
task
  -- based on the number of wheels
        if haswheels then
            if numwheels <= 4 then
                moveAlongRouteSubtaskID = vrf:startSubtask("move-
along",
                    {route = "Light"} )
            end
            if numwheels > 4 then
                moveAlongRouteSubtaskID = vrf:startSubtask("move-
along",
                    {route = "Heavy"} )
            end
  -- if no wheels, assume tracked and send on tracked route.
        else
            moveAlongRouteSubtaskID = vrf:startSubtask("move-along",
                {route = "Tracked"} )
        end
    end
    if vrf:isSubtaskComplete(moveAlongRouteSubtaskID) then
        vrf:endTask(true)
    end
  end
```

In your scenario, create a few ground vehicles. If necessary, specify the number of wheels they have. Give them the Move to Base task and they will pick the correct route to move along.

75.7. Configuring Units

All units have the **parameter-type** `aggregate-object-param`.

In entity-level scenarios, most functionality resides in entities and units have relatively few individual parameters. The principle function of the various elements is to specify controllers and formations, and to specify the **echelon-level** for the different levels of units.

75.8. Configuring Tactical Graphics

The object parameter database includes parameters for tactical graphics. Tactical graphics provided with VR-Forces have the **parameter-type** vrf-base-object-param. You should not change the parameters for the objects provided with VR-Forces. However, if you are using the VR-Forces toolkit, you could create other types of tactical graphics. You would have to include parameters for them in the object parameter database.

75.9. Pathname Interpretation in Simulation Model Set Files

The files that make up a simulation model set, such as *.entity* files and system definition files, usually contain references to other files in the SMS. These supporting files may also have references to other configuration files. VR-Forces supports the file relationships by specifying files using relative paths and by defining directory variables. This allows you to maintain custom simulation model sets outside of the VR-Forces file hierarchy and not have to move them every time you install a new version of VR-Forces.

Examples of directory variables are in the SMS file and OPD file. These lines from the *EntityLevel.sms* file show a binding (model-set-directory) and its use:

```
(simulation-model-set
  (variable-bindings
    (DtRwString
      (model-set-directory  "EntityLevel")
    )
    (DtRwString
      (opd-dir  "$(model-set-directory)")
    )
  ...
  )
  (opd-file  "$(model-set-directory)\vrfSim.opd")
  (physical-world  "$(model-set-directory)\physicalWorldParams.mtl")
  ...
)
```

Similarly, the *vrfSim.opd* file has bindings, which are then used in *.entity* files, as shown.

From *vrfSim.opd*:

```
(variable-bindings
  (DtRwString
  ...
    (system-dir "$(opd-dir)/vrfSim/systems"
    )
    (DtRwString
      (ammoselect-dir "$(opd-dir)/vrfSim/ammoselect"
    )
    ...
)
```

From *M1A2_Tank.entity* (**system-dir** variable):

```
<componentSystem systemName="damage"
  platform="@(system-dir)/damage/ground-heavy-armor.sysdef"/>
```



76. Configuring Sensors

This chapter explains how to configure sensors and sensor signatures.

76.1. Configuring Sensors 1504
- 76.1.1 Adding a New Sensor to a Simulation Object 1504
- 76.1.2 Editing a Sensor System Definition File 1505
- 76.1.3 Specifying a Sensor Signature 1506
- 76.1.4 Specifying Sensor Detection Probabilities 1510
- 76.1.5 Adding Detection Types (Why Doesn't My Sensor Sense Anything?) 1512
- 76.1.6 Adding a New Sensor Domain 1513
- 76.1.7 Illumination and Time of Day 1515
- 76.1.8 Configuring the Effects of Rain on Radar 1516

76.2. Detection Tables 1516

76.1. Configuring Sensors

VR-Forces includes several sensor systems and you can add them to simulation objects using the Simulation Object Editor. This section provides additional details about how sensors are configured. A complete sensor configuration requires:

- One or more **componentSystem** elements in the *.entity* file for sensor systems.
- A list of sensor signature values in the *.entity* file for the sensor domains that can detect the entity type.
- A signature propagator entry in the *./data/simulationModelSets/<model_set>/physicalWorldParams.mtl* file (one per sensor domain).

VR-Forces includes sensors such as radar, infrared, visual, magnetic anomaly detector (MAD), and sonar. The simulation objects configured in the object parameter database are configured with appropriate sensors and sensor signatures. For details about sensor components and how they work, please see API documentation.

76.1.1 Adding a New Sensor to a Simulation Object

It is recommended that you use the Simulation Object Editor to add new sensors to a simulation object. If you want to do it manually, you can edit its *.entity* file.

All sensors provided with VR-Forces are signature-object-sensors. They are distinguished by the domain in which they operate (for example, visual, radar, infrared, MAD, or sonar).

➤ **To add a sensor to a simulation object by editing its *.entity* file:**

1. Copy an existing **componentSystem** element in the *.entity* file and paste it into the file.
2. Change the **systemName** to be unique, for example, sensor-2.
3. Change the name of the system definition file to point to the sensor you want to add.
4. Change the values for the variables that must be specified for the sensor.

Connecting a Sensor

If you add a sensor that has been configured to use the auto-connection feature (all sensors provided by MAK support auto-connections), its connections are handled automatically. (For details, see "75.4.2 Configuring System Connections" on page 1483.)

If you add a new type of sensor, you must connect it to input through its input port and then send the data to its output port for the appropriate controller. The sensor gets input through its `object-types-to-detect` input port. It sends that data out through its `detected-objects` output port. The connections for the visual sensor, in *visual-sensor.sysdef* are:

```
  (connect system:object-types-to-detect visual-sensor:object-types-to-
detect)
  (connect visual-sensor:detected-objects system:detected-objects)
  (connect system:sensor-offset visual-sensor:sensor-offset)
```

If you were to create an acoustic sensor, its connections would look like this:

```
  (connect system:object-types-to-detect acoustic-sensor:object-types-
to-detect)
  (connect acoustic-sensor:detected-objects system:detected-objects)
  (connect system:sensor-offset acoustic-sensor:sensor-offset)
```

76.1.2 Editing a Sensor System Definition File

This example is from the visual sensor system definition file (*visual-sensor.sysdef*):

```
  (visual-sensor
    (component-descriptor-type "signature-sensor-descriptor")
    (component-type  "signature-sensor")
    (min-tick-period -1.000000)
    (min-tick-period-variance 0.100000)
    (periodic-tick-while-paused False)
    (process-state-repository-name "")
    (process-state-repository-type "")
    (is-enabled True)
    (is-enabled-for-clutter-entities False)
    ...
    (detect-only-hostile-forces False)
    (detect-own-force False)
    (detection-types )
    (detect-destroyed-objects False)
    ...
    (sensor-geometry
      (in-range
        (range  $max-range)
      )
    )
    ...
    (sensor-offset  $sensor-position)
    (sensor-domain  "visual")
    (sensor-positional-error  0.000000)
    (sensor-sensitivity $sensor-sensitivity-start)
    (detection-probability
      (round-down 1)
    )
    (detection-level-determinator
      (determinator-type  "signature-detection-level-determinator")
      (detection-level-to-set-hostility  3)
      (detection-level-to-set-radius 4)
      (combat-identification-level-table-file
        (filename "$(detection-dir)\std-visual-detection-table.csv")
      )
    )
   ...
  )
  (controllers )
  (actuators )
```

```
(connections
  (connect system:object-types-to-detect
      visual-sensor:object-types-to-detect)
  (connect visual-sensor:detected-objects system:detected-objects)
  (connect system:sensor-offset visual-sensor:sensor-offset)
)
```

As you can see from viewing the file, most of the parameters that you would want to vary from entity to entity are variables whose values are set in the *.entity* file.

76.1.3 Specifying a Sensor Signature

A sensor signature indicates that a simulation object can be detected by other simulation objects that have a sensor for that signature. The value of the signature represents the maximum distance, at which the entity can be detected. However, this value is affected by modifiers and the calculations made by the propagator. (Technically, the signature does not assume a particular measurement unit. As implemented, distance is measured in kilometers.)

The modifiers specify multipliers for the base value, based on the current state of the object. For example, a simulation object might have a base value of 5.0 for its visual signature, and a modifier of 1.5 when moving. This means the signature of the entity is 7.5 when moving and 5.0 when not moving.

This is the sensor signature block for humans:

```
(sensor-signatures
  (visual
    (base-signature  $visual-signature)
    (modifiers
      (file
        (external-file
          (filename  "signatureRules/lifeform-visual-rules.mtl")
        )
      )
      (file
        (external-file
          (filename  "signatureRules/visual-illumination-rules.mtl")
        )
      )
    )
  )
  (infrared
    (base-signature $infrared-signature)
    (modifiers )
  )
  (radar
    (base-signature $radar-signature)
    (modifiers )
  )
)
```

This sensor-signature block shows that a DI M4 can be detected by visual, infrared, and radar sensors. The base-signature values are variables. Their values come from the *.entity* file.

```
<sensorSignatures>
  <real paramName="visual-signature">1.2</real>
  <real paramName="infrared-signature">1.2</real>
  <real paramName="radar-signature">0.25</real>
</sensorSignatures>
```

The visual sensor has two modifiers. The file *lifeform-visual-rules.mtl* is:

```
(signature-modifiers
  (speed-less-than
    (speed 1.0)
    (multiplier 0.5)
  )
)
```

This indicates that if the entity's speed is less than 1 KMH, multiply the base signature by 0.5. A slowly moving entity is less likely to be seen.

For details about the visual illumination rules modifier, see "76.1.7 Illumination and Time of Day" on page 1515.

If you do not want a simulation object to be detectable by a particular domain of sensor, remove the sensor signature from the *.entity* file. If you want it to be detectable by an additional sensor domain, add its sensor signature to the *.entity* file. You can edit the values of sensor signatures in the Simulation Object Editor. You cannot add sensor signatures in the Simulation Object Editor.

VR-Forces has a set of predefined sensor signatures in *./data/simulationModelSets/EntityLevel/extra/sensor-signatures.csv*. These values are used only in the Simulation Object Editor to quickly set standard sensor signatures for entities in the various sensor domains. The standard sensor signatures are also used if you edit the properties of a sensor system in the Simulation Object Editor.

Table 73 lists the sensor signatures that VR-Forces uses for general categories of simulation objects in *EntityLevel.sms*. These categories do not match the tags list in the Simulation Object Editor or those in the *sensor-signatures.csv* file. They represent a more granular approach to categorizing simulation objects for the purpose of applying a consistent set of sensor signatures.

Table 73. Entity-level sensor signatures

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Military Fixed Wing - stealth technology	2.5	4	6	N/A	N/A	N/A
Military Fixed Wing - Large Jet (Cargo)	7	30	30	N/A	N/A	N/A
Military Fixed Wing - Small Jet (Fighter)	4	6	20	N/A	N/A	N/A
Military Fixed Wing - Large Prop	5	25	15	N/A	N/A	N/A
Military Fixed Wing - Small Prop	4	20	15	N/A	N/A	N/A
Military Fixed Wing - Large UAV (Predator)	5	20	16	N/A	N/A	N/A
Military Fixed Wing - Small UAV	2	2	1.5	N/A	N/A	N/A
Military Fixed Wing - Micro UAV (Hand Launch)	1	0.5	0.5	N/A	N/A	N/A
Civilian Fixed Wing - Large (Passenger)	10	40	35	N/A	N/A	N/A
Civilian Fixed Wing - Medium (regional)	7	30	30	N/A	N/A	N/A
Civilian Fixed Wing - Small (Cesna)	5	20	12	N/A	N/A	N/A
Small Rotary Wing	4	20	20	N/A	N/A	0.3
Medium Rotary Wing	5	25	25	N/A	N/A	0.3
Large Rotary Wing	7	30	30	N/A	N/A	0.3
Military/Civilian Large Cargo Ship (tanker/cargo)	35	40	30	10	8	N/A
Military Aircraft Carrier	35	40	30	10	8	N/A
Military Cruiser	25	30	25	8	6	N/A
Military Destroyer	20	28	25	8	6	N/A
Military Frigate	18	25	25	8	6	N/A
Military Large Patrol (Corvette)	18	25	25	6	4	N/A
Military Small Patrol (covered)	10	25	25	3	2	N/A
Military Small (zodiac)	4	6	8	2	1.5	N/A

Table 73. Entity-level sensor signatures (continued)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Civilian Large Commercial Passenger (Cruise)	45	50	50	10	8	N/A
Civilian Medium Commercial	25	40	35	8	5	N/A
Civilian Small Commercial (tug)	15	30	35	4	3	N/A
Civilian Private Medium	15	25	25	4	3	N/A
Civilian Private Small (24->30ft day cruiser)	10	25	25	3	2	N/A
Civilian Private Micro (jet ski)	5	9	10	1	2	N/A
Civilian Sailboat (J-24)	15	20	10	2	0.5	N/A
Small Lifeboat (bright colors)	7	12	15	1	1	N/A
Large Lifeboat (bright colors)	10	15	15	1	1	N/A
Military Submarine - Large (Ohio)	2	3	2	1	0.01	N/A
Military Submarine - Medium (LA)	1.5	3	2	0.75	0.01	N/A
Military Submarine - Small	1.5	3	2	0.5	0.25	N/A
Military Submarine - Torpedo (non decoy)	0	0	0	1	2	N/A
Human (no camouflage)	1.2	0.25	1.2	N/A	N/A	N/A
Human (with camouflage)	0.5	0.25	0.75	N/A	N/A	N/A
Human in Water (with bright vest)	2	2	1.5	N/A	N/A	N/A
Human in Water (no vest)	0.75	2	1.5	N/A	N/A	N/A
Civilian Passenger Car	4	0.5	4	N/A	N/A	N/A
Civilian Truck/Bus	6	1	6	N/A	N/A	N/A
Military Artillery - large	3	0.75	4	N/A	N/A	N/A
Military Artillery - small	1	0.5	2	N/A	N/A	N/A
Military Large vehicle (Logistics)	5	1	5	N/A	N/A	N/A
Military Medium vehicle (Tank)	3	0.5	4	N/A	N/A	N/A

Table 73. Entity-level sensor signatures (continued)

Entity Category (with example)	Visual	Radar	Infrared	Sonar Active	Sonar Passive	MAD
Military Small vehicle (Jeep/Technical)	1	1	2	N/A	N/A	N/A
Military Micro vehicle (human sized or smaller)	0.5	0.25	1	N/A	N/A	N/A

76.1.4 Specifying Sensor Detection Probabilities

Sensor systems can have a detection probability table. If a sensor system does not have a detection probability table, it always detects targets (assuming they meet detection criteria such as being in line-of-sight, within the detection range, are not being jammed, and so on) and the combat identification level (CID) always advances as the apparent signature increases over time. By adding a detection probability table to a sensor, you can add variability to a sensor's ability to detect targets and update information about them.

The systems provided with VR-Forces do not have detection probability tables.

A detection probability table associates apparent signatures with detection probabilities. (Apparent signature refers to the radar return strength, if the sensor is a radar, or to the apparent brightness of a target on the sensor in question. For more information about apparent signatures, see "76.2. Detection Tables" on page 1516.) The detection probability is applied each detection period of the sensor. For radar, the detection period models the sensor sweep, such that the probability is applied individually for each sensor sweep.

The probability of detection affects detection:

- Initial detection. If an object is not on the current detection list, the probability of detection is applied after the apparent signature is calculated. If a random draw against the probability passes, the target passes to the CID detection table calculation to determine the CID level. If the random draw fails, then the object is treated as if it was not detected. It is processed again on the next sensor detection run.
- Continuing detection. If an object is on the detected list, a random draw is done against the detection probability for its current apparent signature. If this passes, the entity remains detected as normal. Its location and CID level are updated if it is appropriate to do so. If this check fails, the entity remains on the detection list, however the data in the contact information is not updated with any new data, including location, CID level, and so on. On the next sensor detection run, this target is processed again as part of the continuing detection section of the sensor.

The following example shows how you could add a sensor detection probability table to the active-radar-sensor system (*./data/simulationModelSets/EntityLevel/vrfSim/systems/sensors/active-radar-sensor.sysdef*). (The system included with VR-Forces does not have a sensor detection probability table.)

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
      (component-descriptor-type "radar-signature-sensor-descriptor")
      (component-type  "radar-signature-sensor")
      ...
      (detection-period  2.000000)
      ...
      (detection-probability
        (round-down 1)
        (signature 1.0
          (probability 0.2)
        )
        (signature 1.5
          (probability 0.2)
        )
        (signature 2.0
          (probability 0.2)
        )
        (signature 2.5
          (probability 0.2)
        )
        (signature 3.0
          (probability 0.2)
        )
        (signature 3.5
          (probability 0.2)
        )
        (signature 4.0
          (probability 0.2)
        )
      )
      ...
      (detection-period-variance 0.5)
      ...
    )
  )
)
```

76.1.5 Adding Detection Types (Why Doesn't My Sensor Sense Anything?)

To save computation time in VR-Forces, sensors are configured to detect only the objects that the entity needs to know about. If no systems on the entity specify entity types of interest, or the output of the sensor is not connected to anything, the sensor does not attempt to detect simulation objects. In a typical combat entity, a weapon system specifies entity types of interest and a `contact-fusion` controller accepts sensor outputs and creates a contact list. If an entity does not have similar components, then the sensor will not function.

Most simulation objects included with VR-Forces have a `contact-fusion` controller, but objects based on platforms such as *Cultural_Feature.ope* typically do not have this controller. (The *Cultural_Feature_With_Radio.ope* file does have the controller.)

In lieu of a weapon system, you can specify object types for the sensor to detect directly in the sensor.

➤ **To add detection types to a simulation object:**

1. Select the simulation object.
2. For an entity-level simulation object, select the Systems tab.

   For an aggregate-level object, select the Sensors tab.
3. Under Sensors, select the system that you want to adjust and then click Edit System Properties ( ). The properties dialog box opens.
4. Above the Additional Detection Types list, click Add New Object Type ( ). The Add New Object Type dialog box opens.
5. For each field of the simulation object type (Kind, Domain, Country, and so on), select an option from the list.

   The options are drawn from the SISO Enumerations document. As you select an option, the enumeration at the bottom of the dialog box changes to reflect your choice. You do not need to make a selection for each field because VR-Forces can use wild carding to match simulation objects against the final enumeration.
6. If appropriate, add additional enumerations for each simulation object type that is valid.
7. Click OK. You return to the properties dialog box and the object type is added to the list.
8. Click OK.

If your are experiencing an issue where an entity configured with a weapon fails to fire on a simulation object, see "77.2.1 Sensing Targets" on page 1528.

76.1.6 Adding a New Sensor Domain

You can easily add new sensor domains to your simulation objects. Suppose that you want to add an acoustic sensor. You would:

1. Add a block to the signature-propagator-params block in *physicalWorldParams.mtl*.
2. Add a sensor **componentSystem** element to each entity that can detect targets acoustically.
3. Add a sensor signature to each simulation object that can be detected acoustically.

Configuring a Signature Propagator

The *physicalWorldParams.mtl* file specifies the list of sensors available and the propagator to use for each. This is a portion of that file:

```
(physical-world-params
 ...
  (signature-propagator-params
    (radar
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight True)
    )
    (visual
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "visual-propagator")
      (check-line-of-sight True)
     ...
    )
  )
)
```

If you edit the file to add an acoustic sensor, it might look like:

```
(physical-world-params
  (signature-propagator-params
    (visual
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "visual-propagator")
      (check-line-of-sight True)
    )
    (acoustic
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight False)
    )
   ...
  )
)
```

You do not have to be able to see a simulation object to hear it, so you do not need to check line of sight.

Adding a Sensor Component

To add a new sensor component, follow the procedure in "76.1.1 Adding a New Sensor to a Simulation Object" on page 1504. A new acoustic sensor might look like:

```
(acoustic-sensor
  (component-descriptor-type "signature-sensor-descriptor")
    (component-type  "signature-sensor")
    (min-tick-period  2.000000)
    (min-tick-period-variance  0.100000)
    (tick-period-uses-real-time False)
    (process-state-repository-name  "")
    (process-state-repository-type  "")
    (detect-only-hostile-forces False)
    (detection-types )
    (detect-destroyed-objects False)
    (sensor-geometry
      (in-range
        (range  $max-range)
      )
    )
    (sensor-domain  "acoustic")
    (sensor-offset  $sensor-position)
    (sensor-positional-error  0.000000)
    (detection-level-determinator
      (determinator-type  "signature-detection-level-determinator")
      (detection-level-to-set-hostility  3)
      (combat-identification-level-table-file
          "$(detection-dir)\std-acoustic-detection-table.csv")
      )
  )
)
(controllers
)
(actuators
)
(connections
  (connect system:object-types-to-detect
      acoustic-sensor:object-types-to-detect)
  (connect acoustic-sensor:detected-objects system:detected-objects)
)
```



You can call the sensor whatever you want, but the name must not start with a number.

Adding a Sensor Signature

If you add a new sensor, you need to add a signature for it to each simulation object that can be detected by the sensor. If you add an acoustic sensor, a sensor signature might look like:

```
(sensor-signatures
  (visual
    (base-signature  4.000000)
    (modifiers )
  )
  (acoustic
    (base-signature  2.000000)
    (modifiers )
  )
  ...
)
```

You can probably see simulation objects at a greater distance than you can hear them, so the acoustic sensor has a lower base-signature value than the visual sensor.

76.1.7 Illumination and Time of Day

Visual sensors take into account the time of day. They do this through an illumination value that modifies the visual sensor signature. The modification value is configured in *./data/simulationModelSets/<model_set>/vrfSim/signatureRules/visual-illumination-rules.mtl*:

```
(signature-modifiers
  (environmental
    (modifier-coefficients 1.0 0.1)
  )
)
```

"76.1.3 Specifying a Sensor Signature" on page 1506 shows how this file can be referenced in a sensor signature. Given this file, the modifier coefficients are applied to the illumination value:

```
Multiplier = 1.0 * illumination + 0.1
```

The illumination value is derived from the time of day based on parameters in *physicalWorldParams.mtl*:

```
...
  (minimal-illumination-value 0.01)
  (nighttime-illumination-value 0.02)
...
```

where:

- **minimal-illumination-value** indicates the minimal illumination value in the absence of sunlight and moonlight.
- **nighttime-illumination-value** indicates the nighttime illumination value if the moon is completely full.

76.1.8 Configuring the Effects of Rain on Radar

VR-Forces can take into account the effect of rain on radar sensors. It does so based on the amount of rainfall per hour, the range from the sensor to the target, and the **rain-to-attenuation-coefficient** parameter in the radar block of the *physicalWorldParams.mtl* file in an SMS:

```
(radar
  (propagator-descriptor-type "radar-propagator-descriptor")
  (propagator-type "radar-propagator")
  (check-line-of-sight True)
  (rain-to-attenuation-coefficient 0.05)
)
```

The higher the coefficient value, the more that rain reduces the effective range of the radar. Rain is measured in millimeters per hour, where the maximum rain setting in VR-Forces is equal to 20 mm/h.

76.2. Detection Tables

The detection tables determine the level of knowledge that a simulation object has of detected simulation objects. The detection level is used in the Detect Entity condition in plans.

Detection tables are comma-delimited files that are most easily read in a spreadsheet. As illustrated in Figure 632, each column represents a detection interval, in seconds, from 0 through 160. Each row represents an apparent signature from one through 4. (An apparent signature is a simulation object's absolute signature after being modified by factors such as terrain and distance. For details, see the API documentation.)

Figure 632. Detection table displayed in a spreadsheet

	A	B	C	D	E	F	G	H	I	J	K
1	CID Table-radar	Version 1.0		Detection	Interval	(seconds)	->				
2			0	1	5	10	20	40	80	160	
3	Apparent	1	1	1	2	2	2	3	3	3	
4	Signature	1.5	1	1	2	2	3	3	3	3	
5	|	2	1	1	2	2	3	3	3	3	
6	V	2.5	1	2	2	3	3	3	4	4	
7		3	1	2	3	3	3	4	4	4	
8		4	2	3	3	3	4	4	4	4	

The detection level is determined by finding the intersection of the apparent signature and the detection interval. For example, if a simulation object with an apparent signature of 3 has been detected for 40 seconds, its detection certainty level is 4 (full knowledge).

The apparent signature is the ratio of the sensor signature given for the simulation object (in kilometers) to the actual range. So if the apparent signature = 1, the detection interval

shows the increase in combat ID level over time when the target is at the range given by the sensor signature. If the apparent signature is 2, the detection interval shows the detection progression at 1/2 that range, and so on.

The numbers in the table represent the certainty levels:

- 1 – Detected.
- 2 – Classified.
- 3 – Identified.
- 4 – Full Knowledge.

For details about CID detection levels, see "9.4. Detecting Targets" on page 302.



77. Configuring Weapon Systems

This chapter describes configuration files for weapon systems in entity-level scenarios.

77.1. Introduction to Weapon Systems 1520
77.1.1 The Weapon System System Definition Template File 1521
77.1.2 The Weapon System System Definition File 1523
77.2. Configuring Direct Fire Weapons 1527
77.2.1 Sensing Targets 1528
77.2.2 Configuring Ammunition 1529
77.2.3 Hit Probability Tables 1530
77.2.4 Damage Probability Tables 1531
77.3. Configuring Missile Systems 1532
77.3.1 Missile Launcher 1532
77.3.2 Missile Projectile 1533
77.3.3 Laser-Guided Missile Launcher 1533
77.3.4 Laser-Guided Missile Projectile 1534
77.4. Configuring Indirect Fire Weapons 1534
77.5. Configuring Fixed Weapons 1536
77.6. Configuring Bomb Bay Weapon Systems 1536
77.7. Configuring Weapon Systems in the SOE 1537
77.7.1 Specifying Target Types 1537
77.7.2 Configuring Hit Probability 1538
77.7.3 Configuring the Ammunition to Use 1538
77.7.4 Using the Target Configuration Assistant 1539
77.7.5 Configuring the Target Priorities 1541
77.7.6 Configuring the Target Region 1542
77.7.7 Configuring Missile Launcher Systems in the SOE 1542
77.7.8 Configuring Explosive Devices 1543

77.1. Introduction to Weapon Systems

Simulation objects in entity-level scenarios have many weapon systems. They all fall into one of these types:

- Ballistic direct fire, such as rifles and the main gun on a tank.
- Self-propelled or guided munitions, such as missiles and rockets.
- Indirect fire, such as artillery, naval guns, mortars, and hand grenades.
- Fixed munition, such as mines and IEDs.
- Bomb bay. A weapon system for dropping bombs.

You can create and edit weapon systems by editing configuration files in a text editor or in the Simulation Object Editor. The preferred method is to use the SOE. It provides a point-and-click interface, handles all syntax issues, and includes error checking to ensure that all components required by a system are present.

Weapon systems are defined by a system definition template (*.template_sysdef*) file that includes component definitions (controllers, actuators, and sensors) and a system definition file (*.sysdef*) that contains system-specific parameter values and targeting information, such as ammunition, hit probabilities, and target priorities. Typically a weapon system template includes at least one controller that controls aiming and firing and an actuator that models the physical characteristics of the weapon. Some weapons create munitions when they fire, so the engagement process involves an additional simulation object with its own movement system, weapon (warhead) system, or both.

All objects with weapon systems can engage using a task such as Fire At Target, Fire For Effect, Throw Grenade, and so on.

Simulation objects often engage targets autonomously. Objects that engage autonomously require a sensor to detect targets. Weapon systems are configured to engage certain target types, and this information is passed to the sensor so that the appropriate enemy objects can be detected.

Automatic engagements are controlled by a target selection controller, which is a component separate from any one weapon system. This controller is specified in the platform (*.ope*) file. Weapons pass information to and from the target selection controller through a weapon interface. This is a process state repository used to keep track of targets for the weapon and the number of rounds to fire at that target. No port is used, no connections are necessary; it is created for you. The target selection controller keeps a list of all weapon systems' weapon-interfaces, one for each system.

Weapon systems generate the fire interactions that are published on the network for an engagement. Detonation actions may be generated by the weapon or by the munition it launches.

Once there is a detonation, damage is determined by the target. Damage may be configured in one of two ways: by specifying the power of the munition and the damage caused to a target by a detonation of a given power, or by specifying damage probabilities for each munition type against each target type. For more information, see "78.1.1 Calculating Damage from Munitions" on page 1546.

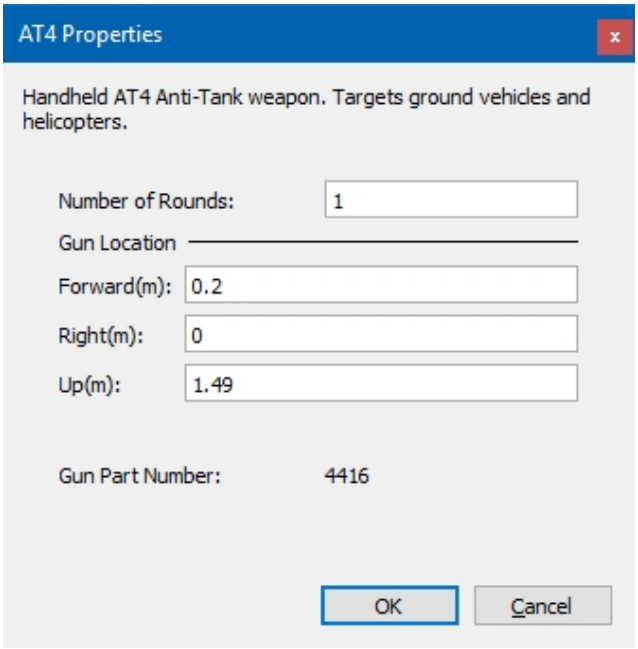
Throughout this chapter, we include snippets from system definition files and other configuration files, but usually without explaining what the various parameters mean. If you want to know more about the parameters, view the referenced systems in the Simulation Object Editor. It has tooltips for most parameters.

77.1.1 The Weapon System System Definition Template File

System definition template files (*.template_sysdef*) contain the sensors, controllers, and actuators that are common to a category of weapon or damage system. They contain variable bindings for attributes that are common to the system type and which are given specific values in the individual weapon system and damage system files. They also contain metadata that is used in the SOE.

The following code is a highly edited version of *handheld-launchers.template.sysdef*. It illustrates how components, resources, and metadata are configured in a template file. Notice that the **meta-data** block includes parameter data for the Gun Location and Gun Part Number parameters. These parameters are editable on a per-entity basis when a handheld launcher system is added to an entity (Figure 633).

Figure 633. Properties for AT4 handheld weapon



The number of rounds parameter is a variable binding in the template file and is given a default value in the system files that use this template, for example, *handheld-AT4.sysdef* (discussed in "77.1.2 The Weapon System System Definition File" on page 1523).

```
(weapon-handheld-AT4
  (controllers
    (human-acquire
      ...
      (target-acquire-criteria $target-acquire-criteria)
```

```
  )
  (main-gun-control
    ...
    (load-ammo-time  $load-ammo-time (default 5.00000))
    (unload-ammo-time  $unload-ammo-time)
    (rounds-per-trigger-pull 1)
    (disabled-by-suppression True)
    ...
    (targeting-control-table $targeting-control-table)
    ...
  )
  (aim-at-point
    ...
  )
  (aim-weapon
    ...
  )
)
(actuators
  (weapon-state
    ...
  )
  (main-gun
    ...
    (art-part-list
      (primary-gun
        (art-part-type $gun-art-part-type)
        (parent-part-type -1)
        (attach-point $attach-point)
        ...
        )
      )
    )
    (neutral-elevation 0.000000)
    (muzzle-speed  $muzzle-speed (default 250.0000))
    ...
    (hit-probability-table $hit-probability-table)
    (range-name  $range-name)
    ...
  )
)
(connections
  ...
)
(resources
  (M84mm
    (resource-type "integer-resource")
    (amount $num-rounds)
    (full-amount $num-rounds)
```

```
    )
  )
  (meta-data
    (system-name  "Handheld Launcher")
    (system-description  "Template for system will allow the
         Simulation Object to be given the ability to add a
         handled launcher to a human.")
    (allowed-state-repository-types "human-param")
    (system-categories "weapon")
    (parameter-data-list
      (art-part-parameter-data
        ...
        (display-label  "Gun Part Number")
        ...
      )
    )
    (meta-data-entry-list )
    ...
  )
)
```

77.1.2 The Weapon System System Definition File

System definition files provide weapon-specific values for the variable bindings in the template file. They reference the template file that they use so that VR-Forces can use the correct components and make the necessary connections for the weapon system. Weapon-specific system definition files also include a targeting control table and a hit probabilities table.

> This section explains the layout of the weapon-specific system definition file for the benefit of advanced VR-Forces users. However, it is recommended that you edit weapon systems using the Simulation Object Editor.

The targeting control table lists each target type that this weapon can fire on and the ammunition types that it can use for that target. If it can use more than one ammunition type, it lists them from the highest to lowest priority. (For more information about targeting control tables, see "77.2.1 Sensing Targets" on page 1528.)

The hit probability table specifies the probability that the weapon can hit a target as a range list or a coefficient list. For more information about hit probability tables, see "77.2.3 Hit Probability Tables" on page 1530.

By default the metadata in a template file is merged with the metadata in the weapon-specific system file for use in the SOE. If you do not want a system file to merge its metadata with the template file, add the **merge-with-sub-systems** variable to the metadata and set it to False, for example:

```
(meta-data
  (system-name "Light Armor damage model")
  ...
  (meta-data-entry-list )
  (ui-filename "")
  (merge-with-sub-systems False)
)
```

This example is an edited version of *handheld-AT4.sysdef*, which references *handheld-launchers.template.sysdef*. Notice how it provides values for the **load-ammo-time** and **unload-ammo-time** variables.

```
(weapon-handheld-AT4
  (systems
    (handheld-launcher
      (variable-bindings
        (DtRwReal
          (load-ammo-time  5.000000)
        )
        (DtRwReal
          (unload-ammo-time  0.000000)
        )
        (DtRwBoolean
          (velocity-affects-hit-chance True)
        )
        (DtRwReal
          (max-skill-level-effect 0.250000)
        )
        (DtTargetAcquireCriteria
          (target-acquire-criteria
            (entity-type 0 0 0 0 0 0 0)
            (time-to-acquire 0.000000)
            (check-maximum-speed-to-acquire True)
            (max-speed-of-target 10.000000)
            (check-elevation-angle-to-acquire False)
            (max-elevation-angle-to-acquire 1.500000)
            (check-elevation-to-acquire False)
            (max-elevation-to-acquire 500.000000)
          )
           ...
          )
        )
        (DtTargetingControlTable
          (targeting-control-table
            (ammunition-list
```

```
      (M84mm
        (amount $num-rounds)
      )
    )
    (entity-target-priorities
      (priority-entry
        (target-type 1 1 -1 1 -1 -1 -1)
        (priorities "M84mm")
      )
      (priority-entry
        (target-type 1 1 -1 -1 -1 -1 -1)
        (priorities "M84mm")
      )
      ...
    )
    (max-targets  1)
    (target-region
      (and
        (not
          (in-range
            (range  30.000000)
          )
        )
        (in-range
          (range  300.000000)
        )
      )
    )
   (fire-with-spot-report-only False)
  ...
 )
)
(DtRwReal
  (muzzle-speed  285.000000)
)
(DtRangeProbabilityTable
  (hit-probability-table
    (entity-range
      (entity-type 1 1 -1 -1 -1 -1 -1)
      (range-determinant
        (step-function
          (range 100.000000
            (probability 0.90000)
          )
          (range 200.000000
            (probability 0.5000)
          )
          (range 500.000000
            (probability 0.30000)
          )
```

```
                    (range 750.000000
                      (probability 0.10000)
                    )
                  )
                )
              )
             ...
            )
          )
        )
        (system-definition
          (filename  "$(system-dir)\weapons\templates\handheld-
launchers.template_sysdef")
        )
        (is-enabled True)
        (part-of-roll-up False)
        (use-parent-system-resource-manager True)
      )
    )
    (connections
      (connect system:all handheld-launcher:all)
      (connect handheld-launcher:all system:all)
    )
    (resources )
    (meta-data
      (system-name  "AT4")
      (system-description  "Handheld AT4 Anti-Tank weapon. Targets
ground
              vehicles and helicopters.")
      (allowed-state-repository-types "human-param")
      (system-categories "weapon" "aimable-weapon")
      (parameter-data-list
        (int-parameter-data
          (parameter-name  "num-rounds")
          (variable-type  "DtRwInt")
          (display-label  "Number of Rounds")
          (display-units  "")
          (source-units  "")
          (list-items "")
          (list-display 0)
          (allow-edit True)
          (display-value True)
          (roll-up-rule "sum")
          (default-value  1)
        )
      )
```

```
    (meta-data-entry-list )
   ...
  )
 ...
)
```

77.2. Configuring Direct Fire Weapons

Entities that have direct fire weapons can fire in response to an engagement task for firing their weapons, and they can also detect targets through their sensors and automatically fire at the targets. Entities have to be able to:

- Know what targets they can fire at.
- Detect the targets.
- Determine the ammunition to use.
- Acquire the target and fire at it.

If you want to create a completely new type of weapon system, you must write code for the sensors, controllers, and actuators that it requires. However, if you need a system that is similar to an existing system, you can often create the new system by copying and modifying the configuration of the existing system. Although you can do so by directly editing configuration files, the recommended way to add and edit weapon systems is to use the Simulation Object Editor. To do so, you must understand how to:

- Configure a weapon system.
- Specify the simulation object types that can use the system.
- Specify the simulation object types that can be damaged by the system.
- Specify the amount of damage a simulation object receives from a munition.

The files that manage these relationships are:

- The weapon system template. The template configures the components common to similar types of weapons, such as ballistic guns or missile launchers.
- The weapon system. Defines a specific system, such as a 30 mm gun, the simulation object types that it can target, ammunition types it can use, hit probabilities, and the simulation object types that can use it.
- The damage system. Specifies the types of munitions that a simulation object is vulnerable to and the damage file to use to determine the probability of damage.

A direct fire weapon:

1. Acquires the target.
2. Selects the ammunition for the target using the ammo select file.
3. Checks that it has that resource.
4. Determines if it hits the target.
5. Sends a fire interaction.
6. After a delay based on distance and muzzle speed, it sends a detonation interaction.

77.2.1 Sensing Targets

Simulation objects that have direct fire weapons will detect targets through their sensors and automatically fire at the targets. Sensor systems can specify what types of targets to detect (using the **detection-types** parameter). The weapon systems can also tell the sensors what they want to target.

You can also edit target types in the Targets tab. See "77.7.1 Specifying Target Types" on page 1537.

Each weapon system has an associated weapon interface. The **targeting-control** parameter takes a list of desired target types and creates a target selection criteria object, which is then set in the weapon interface. This code shows the targeting control for the *120mm-gun.sysdef* file.

```
(DtTargetingControlTable
  (targeting-control-table
    (max-targets  1)
    (target-region
      (in-range
        (range 3000.000000)
      )
    )
    (fire-with-spot-report-only False)
    (ammunition-list
      (M829A1-AP-120mm
        (amount $num-ap-rounds)
      )
      (M830-HEAT-120mm
        (amount $num-heat-rounds)
      )
    )
    (entity-target-priorities
      (priority-entry
        (target-type 1 1 -1 1 -1 -1 -1)
        (priorities "M829A1-AP-120mm" "M830-HEAT-120mm")
      )
      (priority-entry
        (target-type 1 1 -1 2 -1 -1 -1)
        (priorities "M829A1-AP-120mm" "M830-HEAT-120mm")
      )
     ...
    )
  )
)
```

The object's weapon management controller takes the target entity type information from the weapon interfaces and provides it to the sensor systems.

77.2.2 Configuring Ammunition

The targeting control table ammunition-list block specifies the ammunition types that a weapon system can use. Each priority entry in the table specifies a target type (using the best match method) and a space-separated prioritized list of ammunition that should be used for that type of target (highest priority listed first). (The best match method is described in "The Best Match Method" on page 1471. For an example of a targeting control table, see "77.2.1 Sensing Targets" on the previous page.)

The ammunition types that you can include in a weapon system's ammunition-list are specified in *./data/simulationModelSets/<sms>/ammunitionParams.mtl*. The Simulation Object Editor references this file when it displays the lists of ammunition that you can add to a weapon system. Each ammunition type entry in *ammunitionParams.mtl* specifies a set of parameters that define its characteristics and how it is represented in the Simulation Object Editor:

- `munition-type` – 7-digit object type enumeration. This is the object type that is used in the fire and detonation interactions created, and should never contain -1s.
- `tracer-type` – 7-digit object type enumeration. This parameter is optional. Use it if you want a weapon to support tracer rounds.
- `warhead` – VR-Link warhead enumeration value to use in fire and detonation interactions for this munition.
- **guidance-mode**. How the munition is guided, if applicable. Options are:
  - 0 - None.
  - 1 - Laser.
  - 2 - Optical Wire.
  - 3 - Radar.
  - 4 - Thermal.
- **gui-label**. The string used to identify the munition in the Simulation Object Editor.
- **targets**. Object type enumerations for the simulation objects that the weapon can target.
- **tags**. The tags used for filtering the munition in the Simulation Object Editor object lists.

This code shows two entries in *ammunitionParams.mtl*, a missile and a ballistic munition:

```
(spiral-missile
  (munition-type 2 2 222 1 8 0 0)
  (warhead  0)
  (guidance-mode  0)
  (gui-label  "AT-6 Spiral Missile")
  (targets
    (target-type 1 1 -1 -1 -1 -1 -1)
    (target-type 1 3 -1 11 1 -1 -1)
  )
  (tags "3D Model" "Military" "Munition")
)
```

```
(AT-30mm
  (munition-type 2 2 222 2 2 0 0)
  (tracer-type 2 2 222 2 2 0 0)
  (warhead  0)
  (guidance-mode  0)
  (gui-label  "AT-30mm")
  (targets
    (target-type 1 1 -1 -1 -1 -1 -1)
  )
  (tags "mm gun")
)
```

77.2.3 Hit Probability Tables

A weapon system definition file includes a hit probability table. This table is used by the system's actuator to determine whether it hits a target that it fires upon. A hit probability table has one or more **entity-range** entries. The name of the **entity-range** entry is not used, and can be something descriptive. An **entity-range** entry has the form:

- `entity-type` – 7 digit entity type enumeration, which may use -1s as wildcards.
- `range-determinant` – a range determinant can be either a list of probability-range entries with the name "range-list", or a probability-coefficient list with the name "coefficients".
- `range-list` – a list of probability-range entries. A probability-range entry has a name, "range", a range value (in meters), and a list of probability entries. The entry applies to targets in the range from the last entry to the current one (or from 0.0 in the case of the first entry). Each probability entry has a name that the table user uses to find the probability value it is interested in and the value itself (a real number between 0.0 and 1.0.) The ballistic gun looks for the probability entry with the name "probability".

  - For ranges that are greater than the last specified range, the probability is zero.
  - The range for missiles is the range from the missile to the target, not from the entity that fired the missile.

- `coefficients` – a coefficient probability list is a list of coefficient probability entries. Each entry has a probability name and the coefficients of a standard polynomial that can be used with a given range to determine the specified probability. The order of the polynomial is determined by the number of coefficients provided. For example, an entry of:

  ```
  (probability -5.00000E-008 -300000E-012 1.0000)
  ```

  indicates that the formula to use for a given range *r* would be:

  $(-5.00000E\text{-}008 * r^2) + (-300000E\text{-}012 * r) + 1.0$

Values calculated will be constrained to be between 0.0 and 1.0. Zero values are factored out, for example:

```
(probability -5.00000E-008 0.0000 1.0000)
```

indicates that the formula to use for a given range *r* would be:

$(- 5.00000E\text{-}008 * r^2) + (0.0 * r) + 1.0$

This is an example of the hit probability table for the handheld-AK47 system:

```
(DtRangeProbabilityTable
  (hit-probability-table
    (entity-range
      (entity-type  3 -1 -1 -1 -1 -1 -1)
      (range-determinant
        (range-list
          (range 10.000000
            (probability 1.00000)
          )
          (range 200.000000
            (probability 0.9000)
          )
          (range 550.000000
            (probability 0.70000)
          )
          (range 3600.000000
            (probability 0.01000)
          )
        )
      )
    )
  )
)
```

77.2.4 Damage Probability Tables

The damage actuator uses damage probability tables specified by its component descriptor to determine direct and indirect fire damage. Damage tables are in the damage systems in *./data/simulationModelSets/<sms>/vrfSim/systems/damage*. A damage probability table has a list of surface entries, which provide a sub-table for each surface upon which a round might detonate. The name of the surface entry is the surface name used in the damage model.

For the damage-by-power damage system, VR-Forces has damage probability tables for each power type. For more information, see "Configuring the Damage File for Damage-By-Power" on page 1551.

77.3. Configuring Missile Systems

Missiles are launched by an object's missile launcher system, which creates a missile entity. The missile flies itself and detonates based on the use of a hit probability table or on the type of missile.

If a missile launcher system definition specifies a hit probability table, when the missile is created, VR-Forces checks to see if the object type of the target is in the table. If it is, a random number is drawn and compared to the range-based probability. If it is determined that the missile should hit the target, the missile begins its normal fly out. If it collides with the terrain or another entity before reaching the target, it detonates and does not hit the target. However, if the missile does not collide with anything and hits or passes the target, it automatically counts as a hit. The missile detonates on the entity. The SA-21 missile launcher and Patriot missile launcher use hit probability tables.

If a missile launcher system definition has a hit probability table, but the target object type is not listed, or if the system does not specify a hit probability table, detonation is determined based on the type of missile:

- Direct flight. Missiles fly directly to a target. Accuracy is controlled by their maneuverability parameters.
- Laser guided. Missiles follow a laser designator to the target. If the laser is blocked or disrupted, the missile goes to the last known location of the target.
- Route following. Some missiles, such as cruise missiles, can follow a route to a target and detonate either at the target or at a proximity to the target.



The range for missiles is the range from the missile to the target, not from the entity that fired the missile.

77.3.1 Missile Launcher

The missile launcher for direct flight missiles works as follows:

1. The target selection controller chooses the best target for the weapon system and passes it to the weapon through the weapon's weapon-interface.
2. The weapon system's controller determines the right resource for that target type.
3. The missile resource is returned and the system loads the missile to fire.
4. The entity tries to acquire the target. It sends the desired rotation angle and elevation to acquire the target to the rotating mount actuator and elevation mount actuator.
5. Actuators move according to their slew rate speeds and send how far they went in a tick back to the weapon launcher so it can recalculate the remaining amount. It also publishes the articulated part numbers along with amount of rotation/elevation to the network.
6. When the feedback reaches the max angle off boresight, it can fire on the target. The missile object is created.

7. Firing state is sent to the launcher actuator.
8. The launcher actuator sends a fire interaction.

Parameters in the missile launcher's system affect whether it fires. For example, the Maverick Missile Launcher system includes these parameters on the Attributes tab in the Simulation Object Editor:

- Load Ammo Time
- Unload Ammo Time
- Max Vehicle Speed to Fire
- Max Azimuth Angle Off Boresight
- Max Elevation Angle Off Boresight

77.3.2 Missile Projectile

When the missile is created, it flies as follows:

1. Using the missile's movement parameters, the missile controller (track-entity if the target is an entity, or target-point if the target is a location) determines the values to be sent to the missile-maneuver actuator.
2. The missile-maneuver actuator calculates the new position, velocity, acceleration, and orientation of the missile.
3. The missile-collision actuator then checks to see whether the missile will collide with the terrain or an entity. If so, it sends out a detonation interaction.

Missile parameters affect how it detonates. The following code is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/missile-warhead.sysdef*:

```
(actuators
  (warhead
    ...
    (detonate-on-ground-impact True)
    (detonate-on-water-impact True)
    (detonate-on-entity-impact True)
    ...
    (allow-direct-hit True)
  )
)
```

77.3.3 Laser-Guided Missile Launcher

The missile launcher for laser-guided missiles works as follows:

1. The target selection controller chooses the best target for the weapon system and passes it to the weapon through the weapon's weapon-interface.
2. The weapon system's launcher controller tells the lasing controller the targets.

3. The lasing controller looks for an actively lased target that matches the lasing code. If there are no lased targets, it checks to see whether it can autonomously lase. If it can, it starts to lase the target.
4. Once the launcher controller has a lased target, it tells the movement system to rotate to the target.
5. The launcher controller checks for when the entity has rotated to line up with target, within the fire cone defined for it. It creates the missile object and fire interaction.

77.3.4 Laser-Guided Missile Projectile

Once it is created, a laser-guided missile flies as follows:

1. Using the missile's movement parameters, the missile controller determines the values to be sent to the missile-maneuver actuator.
2. The missile-maneuver actuator calculates the new position, velocity, acceleration, and orientation of the missile.
3. The missile-collision actuator then checks whether the missile will collide with the terrain or an entity. If so, it sends out a detonation interaction.

77.4. Configuring Indirect Fire Weapons

Indirect fire weapon systems use the indirect-fire-weapon-controller. Indirect fire weapons fire in response to an indirect fire task, such as Fire-for-Effect or Throw Grenade. They do not fire on their own, like many direct fire weapons do.

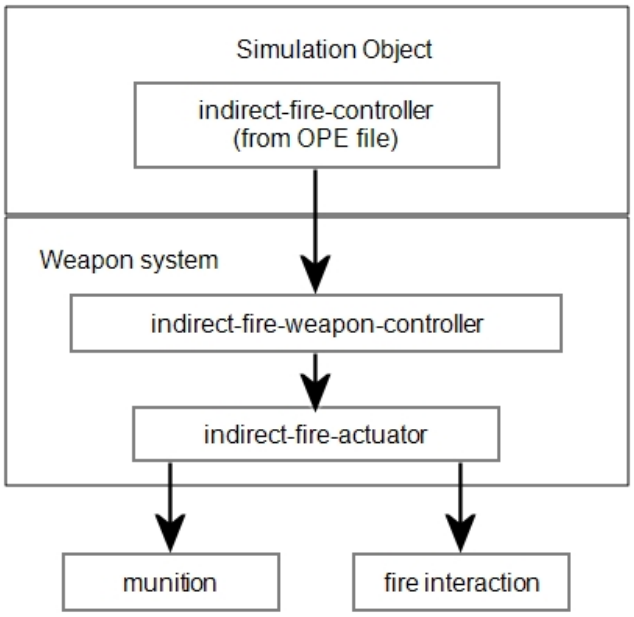
When a simulation object receives an indirect fire task, its indirect-fire-controller queues the task and checks for availability of an indirect fire weapon. It continues to check until a weapon becomes available, then it sends the fire task to the weapon through the indirect weapon interface. If a task requires multiple weapons, the indirect-fire-controller continues this process until the task is complete. (The indirect-fire-controller and indirect weapon interface are analogous to the target-select-controller and weapon interface described for direct fire weapons.)

When an indirect-fire-weapon-controller receives a task, it reads the fire mission from the indirect weapon interface. The indirect fire weapon controller tries to align the turret and load the munition. When the weapon is ready to fire, it passes the information to its indirect-fire-actuator.

The munition is specified as part of the task. The indirect-fire-actuator creates the desired munition and sends out the fire interaction.

Figure 634 illustrates this process.

Figure 634. Indirect fire



Do not confuse the indirect-fire-controller and the indirect-fire-weapon-controller. The indirect-fire-controller is on the simulation object that has the weapon system (it is in the platform OPE file). The indirect-fire-weapon-controller is in the indirect fire weapon system. The controllers are illustrated in Figure 634.

Examples of indirect fire weapons include the M284-155 mm cannon (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/M284-155mm-cannon.sysdef*), the M252-81 mm mortar (*M252-81mm-mortar.sysdef*), and MK45 naval gun (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/MK45NavalGun.sysdef*). Also, although hand grenades would not normally be thought of as indirect fire weapons, the hand grenade tasks call the fire for effect tasks. Therefore, they use the same controllers and actuators as indirect fire weapons.

The munitions used by the indirect fire weapon systems are configured with explosive power in *detonationParams.mtl*. Therefore simulation objects that are susceptible to explosive power can be damaged by them. Their damage system definition files will point to the appropriate damage table files to assess damage from these munitions.

As with other weapon systems, the system definition files for indirect fire weapons have parameters that control when they can be used and how effective they are. You can set the values for these parameters in the Simulation Object Editor on the Attributes and Firing tabs. The Targets tab only has entries for ammunition, since there is no need for hit probability and target selection for indirect weapons.

77.5. Configuring Fixed Weapons

Fixed munitions, such as mines and IEDs, can be triggered:

- Immediately, by a detonate message.
- By proximity of a valid target. The weapon needs a sensor system or a proximity controller that can detect targets.
- Timed detonation, from when the system is armed.

Fixed munitions have a detonation controller. The detonation controller processes set detonation messages to determine how it is to be triggered and whether it is armed. It tells the warhead-detonation-actuator when to arm the weapon. For immediate and timed detonations, it tells the warhead actuator to detonate the device.

For proximity detonations, the detonation controller takes input from the external sensor or the proximity sensor. The warhead-actuator checks for armed and trigger messages from the detonation controller. It also checks for collisions with the terrain and simulation objects. If a collision occurs or it is triggered by the detonation controller, it sends a detonation interaction over the network.

VR-Forces has two types of fixed weapons: naval mines and IEDs.

IEDs are simulation objects that have an explosive device system (*./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/explosive-device.sysdef*). These objects need a sensor to detect targets. The Roadside IED object has a visual sensor. You can specify the ammunition, targets to detect, and the type of fuse on the Targets tab for the system in the Simulation Object Editor.

Naval mines use the *./data/simulationModelSets/EntityLevel/vrfSim/systems/weapons/naval-mine-explosive-device.sysdef* system definition file. It has a proximity sensor built into the system definition, so it does not need an external sensor to detect targets. You can specify the ammunition, targets to detect, and the type of fuse on the Targets tab for the system in the Simulation Object Editor.

You can turn an entity into an IED by adding the explosive device system in the Simulation Object Editor. You can configure cultural features as IEDs by adding the Other Explosive Device system.

77.6. Configuring Bomb Bay Weapon Systems

The bomb-bay weapon system releases bombs in response to Release Bomb tasks. Bombs can drop directly onto targets or be laser guided. Once a bomb is released, it uses the *./data/simulationModelSets/EntityLevel/vrfSim/systems/movement/smart-bomb-dynamics.sysdef* movement system to control movement. It has a warhead actuator that is similar to the warhead actuator in missiles. Its ability to navigate toward and hit its target is controlled by parameters that you can set in the Simulation Object Editor. The Simulation Object Editor also allows you to specify the ammunition type for bomb bay systems. For more information about bombs, see "31.32. Release Bomb Tasks" on page 646.

77.7. Configuring Weapon Systems in the SOE

Like other systems, weapon systems have a General tab for identification information and one or more tabs for physical parameters. Targeting is configured on the Targets tab. When you configure a direct fire weapon system, you must specify:

- The probability that the weapon will hit its target.
- The ammunition it should use.
- Target priorities.
- Target region.

The following sections explain each aspect of target configuration. However, sometimes it can be confusing to understand how to configure the minimum requirements for a target type. The SOE has two features that help you with this process — the Target Configuration Assistant and the Configuration Status window. For details, see "77.7.4 Using the Target Configuration Assistant" on page 1539.

77.7.1 Specifying Target Types

Weapon systems have one or more target types that they can fire upon. You can add, edit, and remove target types. For each target type, you must specify Hit Data. You might also need to specify an ammunition type that is valid for the target.

➤ **To add a target type:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) next to the Target Type label. The Add New Target Type to Hit Table dialog box opens.
4. Specify an object enumeration for this target type.
5. Select a creation option:
   - — Copy Currently Selected. Use the same type of hit table as the currently selected target.
   - — Create Empty Range Table. Create a range list table for configuring how this weapon can fire on the new target.
   - — Create Empty Coefficient Table. Create a coefficient table for configuring how this weapon can fire on the new target.
6. Click OK. The target is added to the Target Type list and a Hit Data table is created based on the selected option.
7. Check the Configuration Status window to see if there are any configuration issues as a result of adding this target type.

77.7.2 Configuring Hit Probability

You can configure hit probability using a range list or a coefficient list. You can configure a range list by editing the table or by editing the graph directly. You configure a coefficient list by editing the table.

➤ **To edit a hit probability table:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Target Type window, select the Target Type that you want to configure.
4. Edit the probabilities as described in "69.9.2 Editing Probability Tables" on page 1371.

Changing the Hit Probability Based on Skill Level

VR-Forces users can change the skill level of entities with the Skill Level set data request. The skill level can be novice, experienced, or expert. The default skill level is expert. If the skill level of an entity is set to novice or experienced, the hit probability of the entity's weapon changes based on the **Maximum Skill Level Effect** parameter for the weapon system. For example, if the **Maximum Skill Level Effect** is set to .20 (20%), then the probability can shift as much as 20% for a novice, 10% for an experienced entity, and 0% for an expert.

For direct fire ballistic weapons, you can set **Maximum Skill Level Effect** on the Attributes tab for the weapon system. For indirect fire weapons (artillery and mortars), you set **Maximum Skill Level Effect** on the Firing tab.

For details about setting the skill level of an entity, see "40.75. Skill Level" on page 900.

77.7.3 Configuring the Ammunition to Use

You can specify the types of ammunition that a weapon system can use. You also specify the default amount. Some weapon systems can use more than one type of ammunition.

➤ **To add a type of ammunition for a weapon system:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Ammunition window. The Choose Ammunition dialog box opens. It lists ammunition types that may be appropriate for this system. Ammunition types that are not supported for this weapon system have an alert icon next to their name. You can add them to the ammunition list, but they will not have any effect on targets unless you also configure them to work with the weapon system.
4. Select the ammunition type that you want to add.
5. Click OK. The ammunition is added to the list in the window.

Adding a New Ammunition Type from the Targets Page

You can add new ammunition types on the Munitions tab. However, you can also start the process from the Choose Ammunition dialog box that you open from the Targets tab.

➤ **To add a new ammunition type:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Ammunition window. The Choose Ammunition dialog box opens.
4. Click Add New from Existing. The Create New Model dialog box opens.
5. Enter a name for the new ammunition type.
6. On the Object Type line, click Edit. The Change Object Type dialog box opens.
7. Specify the object type for the new ammunition by selecting options from the lists or directly entering the object type.
8. Click OK. The new ammunition type is added to the list.
9. Select the Munitions tab.
10. Select the new ammunition type that you added.
11. Specify the parameters for this munition.

Removing an Ammunition Type

➤ **To remove an ammunition type from a weapon system:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Ammunition window, select the row for the ammunition type that you want to remove.
4. Click the Remove button ( ) .

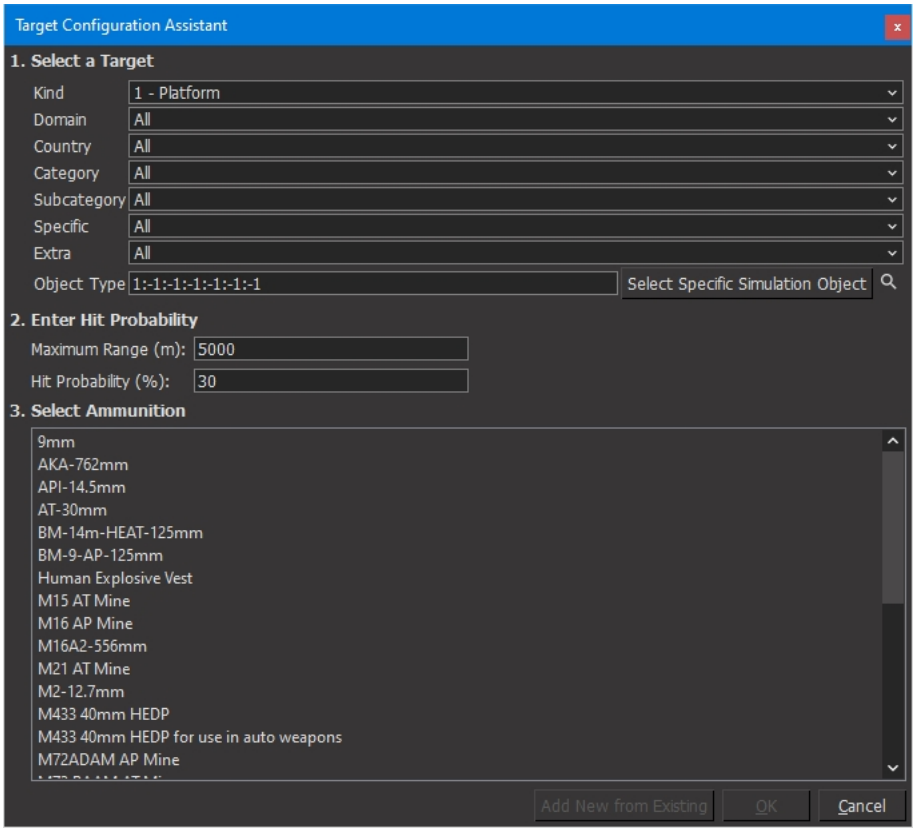
77.7.4 Using the Target Configuration Assistant

You can specify a target type, hit probabilities, and ammunition independently, as described in the previous sections, or you can specify them within one dialog box. This ensures that you have made the minimum configuration necessary for the weapon system to successfully fire on a target type.

➤ **To use the Target Configuration Assistant**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Hit Probabilities section, click the Target Configuration Assistant button. The Target Configuration Assistant opens (Figure 635).

Figure 635. Target Configuration Assistant



4. Select a target type. You can build an object type by selecting options from the lists, directly entering the object type, or selecting a specific simulation object as the target.
5. In the Maximum Range box, enter the maximum distance at which the weapon can hit the target. For range lists, the SOE automatically divides the range for the hit probability table into three segments.
6. In the Hit Probability box, enter the probability of hitting the target at the maximum range.
7. In the ammunition list, select the ammunition to use for this target type. The window lists all possible ammunition types. If an ammunition type is not configured to fire on the target type, it is preceded by an alert icon (as illustrated in Figure 635). You can still add this ammunition type, but you would need to configure it to fire on the target type or it will not work.
8. Click OK. The target type, hit probability, and ammunition are added to the page.
9. Check the Configuration Status window to see if there are any issues for this target type.
10. Optionally, specify weapon target priorities and the target region.

77.7.5 Configuring the Target Priorities

Weapons can only fire at objects that have been configured as possible targets. Within that set of possible targets, you can assign priorities. When the target controller identifies more than one possible target, it chooses its target based on the priorities.

You can specify the types of objects that a weapon can target and the priorities of those target types. For each target type, you can specify the preferred ammunition to use.

➤ **To add a new target priority:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. Click the Add button (+) above the Weapon Target Priorities window. The Add New Target Priority dialog box opens.
4. Specify the object type for the target you want to add.
5. Click OK. The target is added to the bottom of the list.

Changing a Target Priority's Object Type

You can change the object type of a target priority.

➤ **To change the object type of a target priority**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Weapon Target Priorities window, select the target priority you want to change.
4. Click the Edit button (✎) above the Weapon Target Priorities window. The Select Target Type dialog box opens.
5. Change the object type.
6. Click OK.

Changing the Priority of a Weapon Target

Weapon targets are listed in the Weapon Target Priorities window in order of priority, from highest to lowest. You can change the priority of a weapon target by changing the order in which it is listed.

➤ **To change the priority of a weapon target:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Weapon Target Priorities window, select the target whose priority you want to change.
4. Click the Up Arrow above the window to raise the target's priority. Click the Down Arrow to lower its priority.

77.7.6 Configuring the Target Region

You can configure a target region for a weapon system. (Many weapon systems do not have a value for this parameter.) If you specify a target region, the weapon system will not fire at a target unless it is within or without the region. For example, a visual sensor might detect a target at 500 meters, but you do not want the weapon to fire until the target is within 100 meters.

The target region can be specified as a range, a frustum, or both. You can specify a minimum range, maximum range, or both.

➤ **To specify the target region for a weapon system:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. To specify the target region as a range:
    a. Select the Minimum Range check box.
    b. Enter a value for the minimum range.
    c. Select the Maximum Range check box.
    d. Enter a value for the maximum range.
4. To specify the target region as a frustum:
    a. Select the Use Frustum check box.
    b. Specify the maximum and minimum values for the azimuth and elevation.
    c. Optionally, select the Outside Frustum check box, to specify the that the target region is outside of the frustum.

77.7.7 Configuring Missile Launcher Systems in the SOE

VR-Forces has several types of missile launcher systems. The types of missiles that they use determines the parameters that must be set on the Targets tab.

- Laser-guided missile systems configure ammunition (as described in "77.7.3 Configuring the Ammunition to Use" on page 1538) and weapon target priorities (as described in "77.7.5 Configuring the Target Priorities" on the previous page).
- Cruise missile launchers configure ammunition.
- Other missile launcher systems can use hit probability tables or they can calculate hits based on the type of missile, as explained in "77.3. Configuring Missile Systems" on page 1532. When you configure a missile system, you specify how to calculate hit probability by choosing one of these options:
    — Simulate Missile Guidance. Do not use a hit probability table. Calculate based on missile type.
    — Use Probability Table. Configure the launcher like a direct fire weapon system. Specify target type, hit data, ammunition, weapon target priorities, and target region.

77.7.8 Configuring Explosive Devices

Explosive devices, such as IEDs and naval mines, must specify the ammunition type, detonations parameters, and objects to detect.

➤ **To configure an explosive device weapon system:**

1. Select the weapon system you want to edit.
2. Select the Targets tab.
3. In the Detonation Proximity Distance box, specify the distance to use for proximity detonation.
4. Select any of the check boxes that apply for types of impact this device will support.
5. Specify the ammunition, as described in "77.7.3 Configuring the Ammunition to Use" on page 1538.
6. Specify the object types that this device can detect for proximity detonation.



78. Configuring Damage Systems

This chapter describes configuration files for damage systems in entity-level scenarios.

78.1. Introduction to Damage Systems 1546
78.1.1 Calculating Damage from Munitions 1546
78.2. Configuring Damage-by-Power 1546
78.2.1 Configuring the Power of a Detonation 1547
78.2.2 Configuring Detonation Power in the SOE 1550
78.2.3 The Damage File for Damage-By-Power 1551
78.2.4 Adding a New Power Type 1553
78.3. Configuring Damage in the SOE 1555
78.3.1 Configuring Damage Power Entries 1555
78.3.2 Configuring Impact Surfaces 1555
78.3.3 Configuring the Angle of Incidence 1557
78.3.4 Specifying Effect Data 1558
78.4. Viewing Inter-Object Damage Probability Results 1558
78.5. Configuring Damage for Dynamic Terrain 1560
78.5.1 Standard Terrain Damage 1560
78.5.2 High Fidelity Terrain Damage 1561
78.6. Configuring Detonation Results 1562

78.1. Introduction to Damage Systems

Like weapon systems, damage systems are based on high-level template_system files and lower level system definition files that reference the templates. The template files specify the damage actuators and metadata. The system definition files specify the damage to entity surfaces by the different power types that can damage the entity.

You can create and edit damage systems by editing configuration files in a text editor or in the Simulation Object Editor. The preferred method is to use the SOE. It provides a point-and-click interface, handles all syntax issues, and includes error checking to ensure that all components required by a system are present.

Like other systems, damage systems have a General tab and Attributes tab for identification parameters and other default parameters. The damage done to the entity is configured on the Damage tab.

78.1.1 Calculating Damage from Munitions

VR-Forces calculates damage using damage-by-power.

Each munition is configured for its power in the explosive, kinetic, fragmentation, and armor-piercing domains as appropriate. (You can add new domains without writing code.) Damage systems (*.sysdef* files) contain entries for the types of munition power that can affect simulation objects that use the system. They configure the amount of damage that occurs when different levels of power are applied. The damage-by-power model is used for all entity-level simulation objects in VR-Forces 4.5 and later and is the preferred method for calculating damage.

78.2. Configuring Damage-by-Power

In the damage-by-power method of calculating damage, each munition has a `detonation-power-entry` block in *./data/simulationModelSets/base /detonationParams.mtl*. The following is the entry for the 7.62 mm bullet:

```
(detonation-power-entry
  (munition
    ;; 7.62 mm
    (munition-type  2 8 -1 2 2 -1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (kinetic
      (direct 2.24) ;; Kilojoules
    )
  )
)
```

The `munition` block identifies the munition type using the DIS/RPR FOM object enumeration. (The default configuration does not use the **warhead** or **guidance-mode** parameters.)

The `power-list` block is a list of domains and the configuration for a determinant that produces a power value based on an input range. In this example, the 7.62 mm shell has power in the kinetic domain.

The default configurations support these domains:

- Explosive. Munitions that affect simulation objects within some range of the detonation. Explosive munitions represent power in kiloPascals (kPa) of incident pressure.
- Kinetic. Munitions that directly strike a simulation object. Kinetic munitions represent power in kilojoules (kJ) of energy.
- Armor-piercing. Munitions that strike an armored object and penetrate it. Armor piercing munitions represent power in millimeters of penetration.
- Fragmentation. Explosives that cause damage by fragmenting, such as grenades.

You can add other domains without writing code. For details, see "78.2.4 Adding a New Power Type" on page 1553 .

The power values used in the default configuration are based on publicly available information about the power of the various types of munitions and the values used try to be consistent in their relative power.

78.2.1 Configuring the Power of a Detonation

You can configure detonation power by editing *detonationParams.mtl* or in the Simulation Object Editor. This section explains how to edit the file directly and what the parameters mean. For details about editing detonation power in the SOE, see "78.2.2 Configuring Detonation Power in the SOE" on page 1550. To determine the power of a detonation at a specific location a power determinant must be defined. The determinant maps a range value to a power value. Range can have different possible meanings, depending on how the **range-from** parameter is defined for the determinant. If **range-from** is undefined, `impact-point` is assumed unless the determinant does not support this setting.

- If **range-from** is `impact-point`, the determinant determines power based on the range from the impact point of the detonation to some other location in the vicinity. This is used for the explosive and fragmentation domains, but could be used for any power that has some area of effect (for example, perhaps shrapnel or incendiary damage). This is equivalent to collateral damage as defined in the damage-by-ammo-type damage system.

  Example (from *detonationParams.mtl*):

```
(detonation-power-entry
  (munition
     ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
```

```
(power-list
  (explosive
    (range-from "impact-point")
    (range-list
      (range 3.5
        (power 70.0)
      )
      (range 4.6
        (power 40.0)
      )
      (range 5.7
        (power 27.5)
      )
      ...
      (range 35.0
        (power 2.4)
      )
    )
  )
  (kinetic
    (range-from "shooter-to-impact")
    (range-coefficients -2.6600E-004   0.0000   5319.0)
  )
  (armor-piercing
    (range-from "shooter-to-impact")
    (range-coefficients -9.0000E-006   0.0000   180.0)
  )
 )
)
```

- If **range-from** is `shooter-to-impact` (as in the previous example), the determinant determines power based on the range from the entity that fired the round to the point of impact. Any points outside of the point of impact are not affected by the detonation. This is used for the kinetic and armor-piercing domains, but could be used for any power that is directed at a particular target and has no area of effect. This is equivalent to direct damage as defined in the damage-by-ammo-type damage system.
- If **range-from** is `unused`, the determinant determines power without using the input range. This is used by the direct determinant, which ignores the **range-from** setting completely. (For more information, see "Determinant Types" on the facing page.)
- If **range-from** is `shooter-and-impact-point`, the determinant determines power based on both the range from the shooter and from the impact point. This option is not implemented by VR-Forces.

Determinant Types

VR-Forces supports these types of determinant:

- The **range-coefficients** determinant calculates the power based on a series of coefficients. Configuring coefficients works exactly the same way as the hit probability coefficients, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. For example:

```
(kinetic
  (range-from "shooter-to-impact")
  (range-coefficients -1.3000E-003   0.0000   26005.0)
)
```

- The **range-list** determinant determines the power based on a step function. This is configured exactly the same way as the hit probability **range-list**, except that the output value can be any number 0 or greater, not just a probability for 0.0 to 1.0. For example:

```
(explosive
  (range-from "impact-point")
  (range-list
    (range 4.5
      (power 70.0)
    )
    (range 6.0
      (power 40.0)
    )
    ...
    (range 47.0
      (power 2.4)
    )
  )
)
```

- The **direct** determinant always produces the same power at the point of impact, regardless of the range from the shooter. It produces no power at any range from the point of impact. As a result, the **range-from** setting is ignored, because the range input is unused. For example:

```
(kinetic
  (direct 272.873) ;; Kilojoules
)
```

- The **equivalent-tnt-mass** determinant allows you to define an amount of TNT, in kilograms, that produces an equivalent explosion. VR-Forces calculates power at a specified range in kPa. This determinant is used only for the explosive damage category. This example shows its use for M107 HE munition:

```
(detonation-power-entry
  (munition
    ;; M107 HE (155 mm)
    (munition-type  2 9 225 2 14 2 -1)
    (warhead  -1)
```

```
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (equivalent-tnt-mass 5.74)
    )
  )
)
```

78.2.2 Configuring Detonation Power in the SOE

You configure the detonation power of each munition type on its Detonation tab. When you add or edit the detonation power for a munition, VR-Forces creates a copy of the *detonationParams.mtl* in the SMS. It is merged with the version in *base.sms* when a scenario is loaded.

➤ **To configure the detonation power of a munition:**

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. In the Power List, select the power type that you want to configure.
5. If the munition uses a range-list determinant or a coefficient-list determinant, edit the list by editing the table or by directly editing the graph, as described in "69.9. Editing Systems in the Simulation Object Editor" on page 1369.

   If the munition uses the direct determinant, the range is ignored. Specify the direct power in the Direct text box.

   If the munition uses the equivalent TNT mass determinant, specify the explosive power in the Equivalent TNT text box.

Adding a Damage Power Type and Determinant

You can add and remove power types from the power list. When you add a power, you specify a determinant (range list, range coefficient, direct, equivalent TNT mass).

➤ **To add a power and determinant:**

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. On the Power List line, click the Add button (+). The Power and Determinant dialog box opens.
5. In the Categories window, select a power type. The options for this type of power are listed.
6. Select the determinant you want to use.

7. Specify the data for the determinant.
8. Click OK. The power is added to the list.

Changing the Determinant for a Power Type

You can change the determinant for a power type. If you change the determinant the values for the old determinant are removed and you must specify the correct data for the new determinant type.

➤ **To change the determinant for a detonation power type:**

1. Select the Munitions tab of the Object List.
2. Select the munition that you want to configure.
3. Select the Detonation tab.
4. Select the power you want to edit.
5. On the Power List line, click the Edit button (). The Choose Determinant dialog box opens.
6. Select the determinant you want to use.
7. Specify the data for the new determinant.
8. Click OK. The graph and data on the tab change.

78.2.3 The Damage File for Damage-By-Power

Entity damage is configured through the system definition files and the damage files they reference. A system definition file configures power in the **damage-by-munition-power** block.

The system definition files provided with VR-Forces have a `damage-model` block. However none of the files provided use this block. It is available for customers who want to configure munitions using the damage-by-ammo-type method. It also enables backwards compatibility with old SMSs.

Configuring Damage-by-Power in a System Definition File

A system definition file can define any number of `damage-by-power-entry` items, each for a different power-type. This is the damage domain of the power. To add a new type of power, add new settings in *detonationParams.mtl* and in the system definition files for simulation objects that need to be damaged by this effect.

Configuring the Damage File for Damage-By-Power

Damage system files configure damage based on the side of the bounding volume being affected. The damage actuator delivered with VR-Forces uses surfaces of front, rear, left-side, right-side, bottom, and top (other components could model the entity's geometry differently, and therefore use different surface strings). A surface entry has a list of angle-of-incidence entries. An `angle-of-incidence` entry has an angle (in radians).

The `determinant` block specifies the damage. The determinant can be a **step-function** or a list of coefficients. Each determinant can specify probabilities for catastrophic-kill, mobility-kill, firepower-kill or some combination of them.

The *ground-heavy-armor.sysdef* file has a parameter called **round-down** that indicates what probabilities should be used when the input power value is between two steps. For power we want to round down to the next lower power (and lesser probability). Therefore **round-down** is set to 1 (true) for all damage files. If this parameter is not specified, the step function rounds up.

This code, which uses the `step-function` determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/ground-heavy-armor.sysdef*:

```
(damage-by-munition-power
 ...
  (damage-by-power-entry
    (key "explosive")
    (value
      (front
        (angle-of-incidence
          (angle  1.570800)
          (determinant
            (step-function (round-down 1)
              (power 7.500000
                (catastrophic-kill  0.100000)
                (firepower-kill 0.050000)
                (mobility-kill 0.050000)
              )
              (power 12.500000
                (catastrophic-kill 0.500000)
                (firepower-kill 0.250000)
                (mobility-kill 0.250000)
              )
              (power 20.000000
                (catastrophic-kill  0.600000)
                (firepower-kill 0.300000)
                (mobility-kill 0.300000)
              )
              (power 45.000000
                (catastrophic-kill 0.750000)
                (firepower-kill 0.370000)
                (mobility-kill 0.370000)
              )
              (power  70.000000
                (catastrophic-kill 0.990000)
                (firepower-kill 0.490000)
                (mobility-kill 0.490000)
              )
            )
          )
        )
      )
    )
   ...
```

```
      )
    )
  )
```

This code, which uses the **coefficients** determinant, is from *./data/simulationModelSets/EntityLevel/vrfSim/systems/damage/missile-default.sysdef*:

```
(string-DtDamageProbabilityTable-map
  (damage-by-munition-power
    (damage-by-power-entry
      (key "explosive")
      (value
        (front
          (angle-of-incidence
            (angle  1.570800)
            (determinant
              (coefficients
                (catastrophic-kill 0.020000 -0.400000)
                (firepower-kill 0.022000 -0.440000)
                (mobility-kill 0.025000 -0.500000)
              )
            )
          )
        )
    ...
)
```

78.2.4 Adding a New Power Type

To add a new type of power, you add it to the relevant munitions in *detonationParams.mtl*, add it to *./appData/settings/simulationObjectEditor/powerTypes.csv*, and then add it to the appropriate damage system. For example, suppose you want to model the effects of the heat generated by a munition. You could add thermal power. For the sake of this example, suppose that an 85 mm HEAT round has thermal power. To add thermal power, you must:

1. Add the new power to the **detonation-power-entry**:

```
(detonation-power-entry
  (munition
    ;; 85 mm HEAT
    (munition-type  2 2 222 2 8 1 -1)
    (warhead  -1)
    (guidance-mode  0)
  )
  (power-list
    (explosive
      (range-from "impact-point")
      (range-list
        (range 3.5
          (power 70.0)
```

```
          )
          ...
        )
      )
      (kinetic
        (range-from "shooter-to-impact")
        (range-coefficients -2.6600E-004   0.0000   5319.0)
      )
      (armor-piercing
        (range-from "shooter-to-impact")
        (range-coefficients -9.0000E-006   0.0000   180.0)
      )
      (thermal
        (range-from "impact-point")
        (range-list
          (range 3.5
            (power 70.0)
          )
          ...
        )
      )
    )
  )
```

You would have to determine the unit of heat that you wanted to use to measure its effect and come up with realistic power levels at different distances.

2. In the damage systems for the simulation objects that can be affected by thermal power, add a **damage-by-power** entry. Suppose that you want lifeforms to be damaged by thermal power. You would update the *lifeform-default.sysdef* file:

```
(damage-by-power-entry
  (key "thermal")
  (value
    (front-side
      ...
    )
    (left-side
      ...
    )
    (right-side
      ...
    )
    ...
  )
  ...
)
```

3. Add the power type to *./appData/settings/simulationObjectEditor/powerTypes.csv*. The fields for each entry in the file are:

- power. The power name.
- withoutaddition. The text that will be placed on the X and Y axis of graphs.
- withaddition. If there is additional information the Simulation Object Editor gives, the %1 will be replaced with that on the X and Y axis.
- withoutadditionx. If wanted, a different title for the X axis.
- withadditionx. If wanted a different replaceable title for the X axis.
- humanonly. true if this applies only to human damage.
- allowaddition. true if this can be used for addition, false if for title only.
- defaultrangetype. The default range type (shooter-to-impact or impact-point). Please see *detonationParams.mtl* for more information.
- supportedranges. Comma list of supported range calculation values. Please see *detonationParams.mtl* for more information.

78.3. Configuring Damage in the SOE

The Simulation Object Editor provides an easy GUI for specifying damage for each of the power types that apply to an entity. You can configure damage for each surface on an entity and for angles of incidence.

> Most of the damage systems provided with VR-Forces do not configure damage for different angles of incidence.

78.3.1 Configuring Damage Power Entries

The Damage Power Entries window lists the types of damage power that VR-Forces supports. A damage system does not necessarily support all types of damage power. If a damage power type is not supported, it is labeled [Unused]. It does not have entries in the Effect Data table. You can disable use of a damage power type and enable unused damage power types.

➤ To disable a damage power type, click the Remove button ( ).

➤ To enable an unused damage power type, add an entry to its Effect Data table.

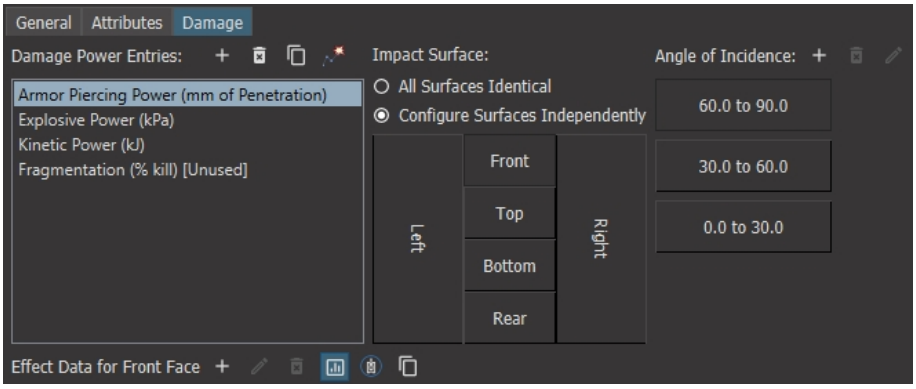
78.3.2 Configuring Impact Surfaces

A damage system can vary the amount of damage sustained based on the surface that is struck or can use the same effect data for all surfaces.

➤ **To configure the same damage for all surfaces of an entity:**

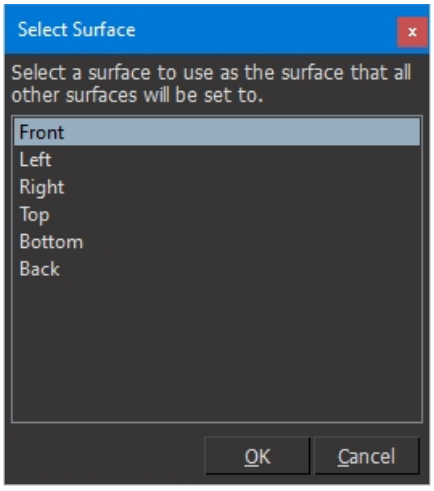
1. In the Damage Power Entries list, select the damage power for which you want to configure impact surfaces.
2. Under the Impact Surface options (Figure 636), select All Surfaces Identical. The Select Surface dialog box opens (Figure 637).

Figure 636. Damage power entries and impact configuration



3. In the Select Surface dialog box, select the surface that will represent all surfaces and then click OK.

Figure 637. Select Surface dialog box



Text displays below the Damage Power Entries list to indicate that you are editing effect data for all surfaces.

4. Optionally, specify the angles of incidence to use, as described in "78.3.3 Configuring the Angle of Incidence" on the facing page.
5. Specify effect data for each angle of incidence, as described in "78.3.4 Specifying Effect Data" on page 1558.

If you configure damage differently for each surface, you must specify the angle of incidence and effect data for each surface.

➤ **To specify damage for each surface:**

1. In the Damage Power Entries list, select the damage power for which you want to configure impact surfaces.

2. In the Impact Surface options list, select Configure Surfaces Independently. Text displays below the Damage Power Entries list to indicate which surface you are editing (Figure 636).
3. For each surface that you want to configure:
    a. Click the box that represents the surface you want to configure.
    b. Optionally, specify the angles of incidence to use for this surface, as described in "78.3.3 Configuring the Angle of Incidence" below.
    c. Specify effect data for each angle of incidence, as described in "78.3.4 Specifying Effect Data" on the next page.

78.3.3 Configuring the Angle of Incidence

You can specify different damage results based on the angle at which a munition strikes an entity. If a damage system treats all surfaces identically, you configure one set of angles of incidence. If it treats different surfaces differently, you can configure angles of incidence for each surface. The default for most damage systems is that there is no difference based on the angle. You can specify multiple angles as ranges. The total range of all angles must be 0 degrees to 90 degrees.

➤ **To add an angle of incidence:**

1. In the Damage Power Entries list, select the damage power for which you want to configure angles of incidence.
2. If you are configuring each impact surface independently, click the surface for which you want to configure the angle of incidence.
3. Click the Add button (+) next to the Angle of Incidence label (Figure 636 on the previous page). The New Angle dialog box opens.
4. Enter an angle, in degrees.
5. Click OK. The new angle is added to the list.

Editing an Angle of Incidence

➤ **To edit an angle of incidence:**

1. In the Damage Power Entries list, select the damage power for which you want to configure angles of incidence.
2. If you are configuring each impact surface independently, click the surface for which you want to configure the angle of incidence.
3. Click the angle that you want to edit.
4. Click the Edit button (✎) next to the Angle of Incidence label. The Edit Angle dialog box opens.
5. Enter the new angle.
6. Click OK.

78.3.4 Specifying Effect Data

For each of the damage power types used by a damage system, you can specify effect data. Effect data can be specified using a range list or a coefficient list. You can change the type of list that is used for a damage system. You can specify effect data for catastrophic kill, firepower kill, and mobility kill. (Many of the systems provided with VR-Forces only configure catastrophic kill.)

If you are using a range list, edit the ranges and range values as described in Figure 567 on page 1372.

If you are using a coefficient list, edit the probability table as described in "Editing a Coefficient List Probability Table" on page 1373.

78.4. Viewing Inter-Object Damage Probability Results

Sometimes when VR-Forces users create a new weapon system or damage system, the systems do not work as expected. The system might not fire on a target or might not damage it when it does. This is usually due to a mismatch in systems or a misunderstanding of how damage probabilities are calculated. The Damage Probabilities viewer shows you what you can expect when a specific entity fires on another specific entity.

➤ **To view damage probability results:**

1. Choose **View > Damage Probability Results**. The Damage Probabilities viewer opens (Figure 638).

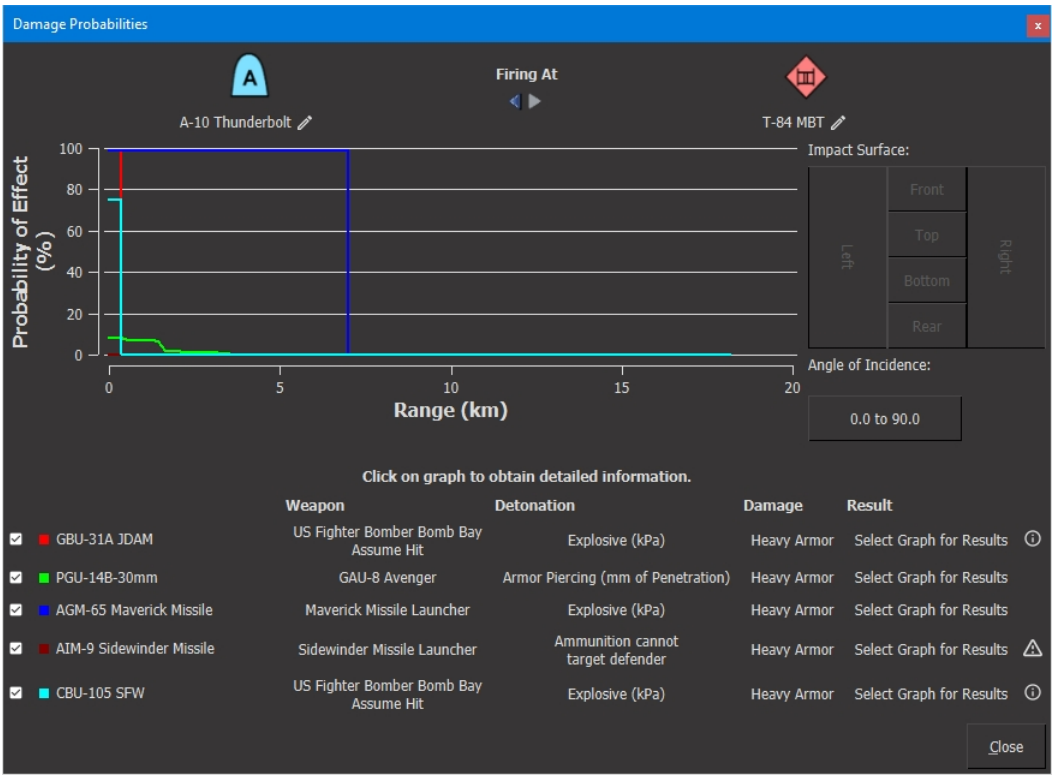
Figure 638. Damage Probabilities viewer



2. Click the Select Attacker button. The Select Simulation Object dialog box opens. It is an object list similar to the object list in the SOE or the Simulation Objects Panel.
3. Select the entity that you want to test as the attacker.
4. Click OK. The entity name is displayed under the button.
5. Click the Select Defender button. The Select Simulation Object dialog box opens.
6. Select the entity that you want the attacker to fire on.

7. Click OK. The entity name is displayed under the button. The viewer redisplays to show the probability graph for each weapon that the attacker has (Figure 639).

Figure 639. Damage Probabilities viewer with entities selected



8. Click in the graph at the range for which you want to see the probability of effect result. The Information table title (Information at range x.x) shows the range you have selected. The Result column in the Information table shows the probability that each weapon system on the attacker has of damaging the defender.
9. If the defender has multiple impact surfaces configured, select the surface whose probability you want to view.
10. If the defender has multiple angles of incidence configured, select the angle of incidence whose probability you want to view.
11. Optionally, clear the check boxes next to individual weapon systems to filter the number of graphs in the window.
12. Optionally, to reverse the relationship between attacker and defender, click the Firing At arrow buttons.
13. Optionally, to see the configuration page for the attacker or defender, click the Edit button () next to the entity name.
14. Optionally to see the configuration page for the weapon, munition, or damage system, click the label in the appropriate column.

15. Optionally, click an Alert button (⚠) or Information button (ⓘ) to display explanatory information, such as the fact that a weapon system cannot target the selected defender.

78.5. Configuring Damage for Dynamic Terrain

Dynamic terrain damage is configured using the damage-by-power approach. It is configured in *./data/simulationModelSets/base/vrfSim/systems/other/dynamic-terrain.sysdef* in the settings for the dynamic-terrain-munition-damage controller. Terrain can be damaged using standard terrain damage or high fidelity terrain damage.

ⓘ Dynamic terrain is supported on OpenFlight and FBX terrain patches, and Heightfields and Point Model Replace in osgEarth. MetaFlight, Terrapage, and CDB terrains, and also props do not support dynamic terrain.

78.5.1 Standard Terrain Damage

When you use standard terrain damage, the controller publishes a series of concentric damage areas of decreasing damage from the impact point. The type of terrain damage (that is, the switch in the terrain to be updated) is defined by **standard-terrain-damage-type** in *dynamic-terrain.sysdef*. The detonation power domain that causes this damage is defined by **standard-power-type**. The `standard-terrain-damage-table` block specifies the concentric areas. When a detonation is received, for each entry in this table a new area is created based on the range that produces the specified power. Then the switch in the terrain model is set to the specified damage setting.

This code is from *./data/simulationModelSets/base/vrfSim/systems/other /dynamic-terrain.sysdef*:

```
(standard-terrain-damage-table
  (damage-by-power-entry
    (power 15)  ;; explosive power is in kPa
    (damage "slight")
  )
  (damage-by-power-entry
    (power 40)
    (damage "moderate")
  )
  (damage-by-power-entry
    (power 70)
    (damage "destroyed")
  )
)
```

This table generates three dynamic terrain areas. The smallest area is defined by the area on the terrain that experiences an explosive power of at least 75. It is switched to a value of "destroyed". The next larger area is the area that experiences power between 40 and 75. It is switched to a value of "moderate". Finally the outer area is the area that experiences power between 15 and 40. It is switched to a value of "slight".

78.5.2 High Fidelity Terrain Damage

You can configure high fidelity damage areas along with the standard damage. To determine where high fidelity damage areas should be created, VR-Forces queries the terrain to find any dynamic terrain features that match the given **dynamic-terrain-type** setting. This can be used to generate additional or more specific types of damage to select locations in the terrain that support a higher fidelity of damage. Just as with standard damage, a **power-type** is specified that corresponds to the detonation power domain that is causing this damage. High fidelity terrain damage is configured in the `high-fidelity-terrain-damage-table` block. It has the same format as the `standard-terrain-damage-table`.

The high fidelity damage lets you specify the damage-algorithm used. VR-Forces only supports the range-based algorithm, but you can develop your own algorithms in a plug-in. If you add additional damage algorithms, they might require configuration options that are not supported in the `high-fidelity-damage-table` provided by MAK. If necessary, you can specify additional parameters using the **additional-algorithm-config** parameter, which accepts a list of key/value string pairs.

High fidelity damage can produce a secondary detonation. This can be specified by setting **detonation-on-destroy** to True and defining the **detonation-munition** parameter. The **delay** parameter defines a delay in seconds between when power was first experienced and when the damage should be done. This allows the secondary detonation to create a particle effect that can mask the underlying terrain switch.

This code is from *./data/simulationModelSets/base/vrfSim/systems /other/dynamic-terrain.sysdef*:

```
(high-fidelity-terrain-damage-table
  (entry
    (dynamic-terrain-type "damage_structure")
    (power-type "explosive")
    (damage-algorithm "range-based")
    (delay 2.0)
    (detonate-on-destroy True)
    (detonation-munition 2 0 0 4 2 0 0)
    (terrain-damage-table
      (damage-by-power-entry
        (power 15)  ;; explosive power is in kPa
        (damage "slight")
      )
      (damage-by-power-entry
        (power 40)
        (damage "moderate")
      )
      (damage-by-power-entry
        (power 70)
        (damage "destroyed")
      )
    )
    (additional-algorithm-config
    ;; Additional key/value pairs (string values) can be added here
```

```
      ;; for custom algorithm configuration
      ;; For example:
      ;; (my-custom-algorithm-config "config value string")
      )
   )
)
```

78.6. Configuring Detonation Results

The SISO Enumerations Document specifies a list of detonation results, such as Entity Impact, Ground Impact, Detonation, and Air Burst. When a munition detonates, a detonation-result message is sent on the network. Applications can use this information to determine how to visualize the detonation or for other purposes.

By default, VR-Forces uses a small subset of the detonation results specified in the SISO Enumerations Document. You can customize the detonation result messages sent by VR-Forces by editing the detonation-results-table in *./data/simulationModelSets /base/detonationParams.mtl*.

For each munition type, you can configure the detonation results for entity impacts, surface impacts, and mid-air detonations. You can specify the detonation results using the strings listed in the comments at the top of the detonation-results-table or you can use any of the integer values specified in the SISO Enumerations Document for the different detonation result types. The *DtDetonationManager* converts the strings or integers into the appropriate detonation result messages to send over the network. (For details about the *DtDetonationManager*, please see the *VR-Forces Developers Guide* class documentation. If editing *detonationParams.mtl* does not meet your needs, you can use the API to override *DtDetonationManager*.)

This code shows the entry in the detonation-results-table for an anti-ship missile:

```
(detonation-result-entry
   (munition
      ;; Anti-Ship Missile (incl. Exocet)
     (munition-type  2 6 -1 1 -1 -1 -1)
     (warhead  -1)
     (guidance-mode  0)
   )
   (entity-impact-result-list
      (default-entity
         (entity-type -1 -1 -1 -1 -1 -1 -1)
         (detonation-result "entity-impact")
      )
   )
   (surface-impact-result-list
      (default
         (surface-type "default")
         (detonation-result "ground-impact")
      )
      (water
         (surface-type "water")
```

```
            (detonation-result "detonation")
         )
         (undefined
            (surface-type "undefined")
            (detonation-result "detonation")
         )
         (ground
            (surface-type "ground")
            (detonation-result "ground-impact")
         )
      )
      (mid-air-detonation-result "detonation")
)
```



79. Configuring Movement Systems

This chapter describes configuration files that affect simulation object movement.

79.1. Configuring Formations 1566
79.1.1 Formation Files 1566
79.1.2 Creating a User-Defined Formation 1567
79.2. Configuring Obstruction Avoidance 1568
79.2.1 Configuring Object Types 1568
79.2.2 Configuring Vector Feature Types 1568
79.3. Configuring Embarkation 1569

79.1. Configuring Formations

> Although you can create and edit formation files by hand as described in this section, the recommended approach for create and edit formations is to use the Simulation Object Editor.

The object parameter database lists the formations available to a unit in the `formations` element in its *.entity* file. Each formation is identified by its name and the path to the formation file, for example:

```
<formations paramName="formation-list" autoLayout="0">
  <formation formationName="Line" formationFile=
      "@(formation-dir)/platoonLine.frm"/>
  <formation formationName="Column" formationFile=
      "@(formation-dir)/platoonColumn.frm"/>
  <formation formationName="Wedge" formationFile=
      "@(formation-dir)/platoonWedge.frm"/>
  <formation formationName="Vee" formationFile=
      "@(formation-dir)/platoonVee.frm"/>
</formations>
```

79.1.1 Formation Files

A formation file specifies the name of the formation and has an entry for each simulation object in the default formation definition. Each entry specifies a **promotion-id**, a **leader-promotion-id**, and a **position-offset**, where:

- The **promotion-id** specifies the order in which simulation objects are promoted to unit leader if the leader is incapacitated.
- The **leader-promotion-id** specifies the unit member from whom the entry's position will be offset.
- The **position-offset** specifies how many meters behind, to the right, and above the entry will follow its leader. The **position-offset** is specified in terms of the body coordinate system of its leader.

The following is a portion of the formation file for a column formation:

```
(column-formation
  (entry-0
    (promotion-id  0)
    (leader-promotion-id  -1)
    (position-offset  0.000000 0.000000 0.000000)
  )
  (entry-1
    (promotion-id  1)
    (leader-promotion-id  0)
    (position-offset  -25.000000 0.000000 0.000000)
  )
  (entry-2
    (promotion-id  2)
```

```
    (leader-promotion-id  1)
    (position-offset  -25.000000 0.000000 0.000000)
  )
  (entry-3
    (promotion-id  3)
    (leader-promotion-id  2)
    (position-offset  -25.000000 0.000000 0.000000)
  )
)
```

In a column, each simulation object follows the simulation object in front of it by 25 meters. Therefore, the **leader-promotion-id** for each entry is the **promotion-id** of the simulation object in front of it, and each entry has a **position-offset** of -25 meters.

The number of entries in a formation file is finite. However, it is possible to place more simulation objects into a unit than there are entries in a formation file. If this happens, VR-Forces increments the **promotion-id** of the last entry in the formation file and assigns the new simulation object the same **position-offset** as the last entry in the formation file. For a simple formation like a column, this works well. If you are using a more complex formation, this mechanism might create unintended relationships among simulation objects.

If you want to change the layout of the formations provided with VR-Forces, you can edit the formation files.

79.1.2 Creating a User-Defined Formation

You can create formations other than those provided by VR-Forces. It is recommended that you create formations in the Simulation Object Editor. If you use the Simulation Object Editor, you do not have to edit any formation files or *.entity* files by hand.

➤ **To create a formation:**

1. Using one of the formation files supplied with VR-Forces as your template, place as many entries as you want in the formation file and specify their offsets.
2. Save the new formation under a new name.
3. Add an entry to the formation list block in the *.entity* file for each unit that you want to be able to use the formation. Be sure to use the same structure and naming conventions as the default entries. For example:

```
<formations paramName="formation-list" autoLayout="0">
  <formation formationName="Line" formationFile="@(
    formation-dir)/DI-companyLine.frm"/>
  <formation formationName="Column" formationFile="@(
    formation-dir)/DI-companyColumn.frm.frm"/>
  <formation formationName="Wedge" formationFile="@(
    formation-dir)/DI-companyWedge.frm"/>
  <formation formationName="Vee" formationFile="@(
    formation-dir)/DI-companyVee.frm"/>
</formations>
```

We recommend that you configure your formations so that simulation objects follow the leader, rather than the next closest member of the formation (as is done in column formation.) This ensures that formations have the best chance of maintaining their formation as they maneuver.

79.2. Configuring Obstruction Avoidance

You can configure ground entities to avoid specific entity types, vector features, or both.

> The more vector feature types and entity types specified for avoidance, the more memory and processing power required.

Obstruction avoidance is influenced by the **stopping-distance-factor** and **lateral-clearance** parameters. If entities display erratic behavior, try raising the values, lowering the speed of the entities, or both.

79.2.1 Configuring Object Types

To specify the types of objects that an entity type should avoid, add or remove the appropriate object type from the obstruction sensor entry in the movement system definition file. For example, these object types are configured for ground entities:

```
(object-type
  (object-type  1 (18 -1 -1 2 -1 -1 -1)) ;; vrfObstacles object type
  (object-type  1 (1 -1 -1 -1 -1 -1 -1)) ;; DtPlatform entity kind
  (object-type  1 (3 -1 -1 -1 -1 -1 -1)) ;; DtLifeForm entity kind
)
```

79.2.2 Configuring Vector Feature Types

Configuring avoidance of vector features relies on the attributes on the feature data as well as the named queries defined in *./appData/settings/featureconfig.txt*. To configure specific feature data for obstruction avoidance, modify the feature-types-to-avoid type list in the obstruction sensor in the appropriate system definition file. The types in this file are named feature queries, and you can add as many as you need.

Example of *human.sysdef*:

```
(obstruction-sensor
  ...
  (feature-types-to-avoid
    (type "MAK_OBSTACLE")
  )
)
```

Example of *air-cushion-default.sysdef*. This entity specifically calls out building, infrastructure, and vegetation, instead of obstacle, because obstacle also includes waterway.

```
(obstruction-sensor
  ...
  (feature-types-to-avoid
```

```
      (type "MAK_BUILDING")
      (type "MAK_INFRASTRUCTURE")
      (type "MAK_VEGETATION")
    )
  )
```

79.3. Configuring Embarkation

Entities and units that are configured to allow entities to embark on them have an occupancy-director-controller component. Its parameters configure embarkation:

- The locations at which entities can embark on an entity are configured as **ingress-points**.
- The locations from which entities can disembark are configured as **egress-points**.
- The number of entities that can embark on an entity is defined by the **embarkation-slots** parameter.

The **embarkation-slots** parameter has one or more **embarkation-slot-entry** parameters that identify a location on the entity where an embarked entity can be placed. The **embarkation-slot-entry** parameters also specify the orientation of the embarked entity and the types of entities that can occupy the slot.

The **ingress-points** and **egress-points** parameters each have one or more **load-entry** parameters. The **load-entry** parameters specify one or more **load-entry-point** parameters. If there is one **load-entry-point**, an embarked entity enters the parent entity at that point and is immediately placed in an available slot. If a **load-entry** has multiple load entry points, they specify the points in a route that the embarking entity follows to reach its slot. This configuration is for large parent entities, such as aircraft carriers, on which entities land and then taxi to a slot.

You configure embarkation in the Simulation Object Editor. For details, see "Chapter 73. Configuring Embarkation" on page 1441.



80. Configuring Emitters and Radios

This chapter describes configuration files for emitters and radios. For information about configuring the display of emitters, see "Chapter 56. Configuring Emitter Volumes" on page 1135.

80.1. Configuring Emitters 1572
80.2. Configuring Radios 1574
80.2.1 Publishing Radio Transmitters 1574

80.1. Configuring Emitters

All fixed-wing aircraft have emitters as part of their radar sensor. Emitters have search modes. Search modes have a list of beams. You can configure entities with emitters that have more than one beam. You can also configure multiple emitters. The basic radar sensor with an emitter and two search modes (*./data/simulationModelSets/<model_set>/vrfSim/systems/sensors/active-radar-sensor.sysdef*) is as follows:

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
       (component-descriptor-type "radar-signature-sensor-
           descriptor")
       (component-type  "radar-signature-sensor")
       . . .
       (sensor-domain  "radar")
       (sensor-offset $sensor-position)
       (sensor-positional-error  0.000000)
       (detection-level-determinator
          (determinator-type  "signature-detection-level-
              determinator")
          (detection-level-to-set-hostility  3)
          (combat-identification-level-table-file
             (filename  "$(detection-dir)\std-radar-detection-
                 table.csv")
          )
       )
       (detection-period  0.000000)
       (publish-emitter-system True)
       (emitter-system
         (system-name-enum  5570)
         (system-function-enum  5)
         (radar-mode-list
           (search-mode
              (index  0)
              (radar-mode  "Search")
              (beam-list
                (beam-0
                (index  0)
                (type  "track")
                (frequency  4999999488.000000)
                (frequency-range  10000.000000)
                (power  70.000000)
                (pulse-repetition-frequency  0.000000)
                (pulse-width  0.000000)
                (azimuth-center  0.000000)
                (azimuth-sweep  0.523599)
                (elevation-center  0.349066)
                (elevation-sweep  0.523599)
```

```
                (sweep-sync  0.000000)
                (beam-function-enum  1)
                (beam-param-index  0)
              )
            )
          )
          (track-mode
            (index  1)
            (radar-mode  "Track")
            (beam-list
              (beam-0
               ...
              )
            )
          )
        )
      )
    )
  )
```

To configure multiple emitters for an entity, add additional radar-sensor blocks. Each block must have a unique name. For example:

```
(radar-sensor-1
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type  "radar-signature-sensor")
  . . .
  (emitter-system
    ...
    )
  )
)
(radar-sensor-2
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type  "radar-signature-sensor")
  . . .
  (emitter-system
    ...
    )
  )
)
```

To add multiple modes on an emitter, add additional mode blocks to the radar mode list. The names of radar modes (for example, track-mode and search-mode) and the value for the **radar-mode** parameter are arbitrary. You can name them whatever you want.

To add multiple beams on a radar mode, add additional **beam-list** blocks and change the name of each block. For example:

```
(search-mode
  (index  0)
```

```
    (radar-mode  "Search")
    (beam-list
      (beam-0
        ...
      )
      (beam-1
        ...
      )
    )
  )
```

80.2. Configuring Radios

All radios are *DtVrfRadio* classes or subclasses. The radio is responsible for managing incoming and outgoing radio messages on a single simulation object. Each radio is associated with a specific communication model, and uses this communication model to send and receive messages.

Radios are configured in platform (OPE) files. The radios block in the OPE file lists all the radios on the simulation object and the parameters for each radio. Parameters include the type of radio descriptor and type of radio class, which are strings used to create the C++ classes from the factories. The **comm-model-name** parameter identifies which communication model a radio is associated with. Every radio must be associated with a communication model.

A typical radio entry is:

```
(main-radio
  (radio-descriptor-type "radio-descriptor")
  (radio-type  "default-radio")
  (comm-model-name  "default-radio-model")
  (publish-transmitter True)
  (transmitter-params
    (radio-type  7 1 225 2 1 20)
    (initial-frequency  3000000.000000)
    (power  30.000000)
  )
)
```

80.2.1 Publishing Radio Transmitters

The external communication model requires that any simulation objects sending or receiving messages though the communications effects server publish a DIS or HLA radio transmitter object. By default, the internal communication model does not publish radio transmitters.

Publishing radio transmitters is controlled by two parameters:

- The **publish-transmitters** parameter in *./data/simulationModelSets/<model_set>/commModelParams.mtl*. By default, this parameter is False for the simple communication model and True for the external communication model.

- The **publish-transmitter** parameter in an entity's OPE file. This parameter is set to true for all object types.

To publish radio transmitters, both of these parameters must be True. If you do not change the OPE files and set **publish-transmitters** to True, all simulation objects will publish radio transmitters. If you set **publish-transmitters** to True and do not want all simulation objects to publish radio transmitters, you can set **publish-transmitter** to False for the object types that you do not want to publish them.



81. Joystick and Keyboard Controls

This chapter explains how to configure joysticks and the keyboard to control entity movement.

> Out of the box, VR-Forces has minimal joystick support. To learn more about joystick controls, see the Joystick Controller sim engine example (exampleJoystick) in the *VR-Forces Developers Guide*.

81.1. Using a Joystick to Control Entities 1578
81.1.1 Switching Between Systems 1578
81.1.2 Disabling Joystick Control 1579
81.2. Using the Keyboard to Control an Entity 1579
81.3. Configuring Joysticks and Keyboard Control 1579
81.3.1 Creating a Joystick Configuration 1581
81.3.2 Mapping the Joystick Stick 1582
81.3.3 Mapping Joystick Buttons and Keyboard Keys 1583
81.3.4 Configuring “Switch Between” Options 1585
81.3.5 Using Multiple Joysticks 1588
81.3.6 Specifying a Favorite Joystick Configuration 1588
81.4. Installing a Gamepad on Linux 1589

81.1. Using a Joystick to Control Entities

VR-Forces supports use of joysticks, other types of game controllers, and keyboard keys to control ground vehicle entities. You can attach multiple joysticks, which allows you to control multiple entities. You can also use a joystick and keyboard mappings together to control an entity.

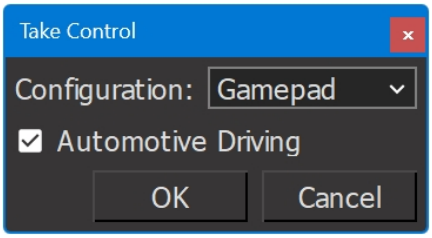
Joystick control is managed in the GUI. Joysticks should be attached to the computer from which you are running the GUI. Control messages are sent to the sim engine, which can be on any computer in the simulation.

- You cannot enable or disable joystick support for all entities at once. You can only enable joysticks for individual entities.
- Embarked entities controlled by joystick cannot disembark.
- For information about how to configure joysticks, see "81.3. Configuring Joysticks and Keyboard Control" on the facing page.

➤ **To enable joystick control:**

1. Select the entity for which you want to enable joysticks.
2. Choose **Objects > Joystick > Take Control**. The Take Control dialog box opens.

Figure 640. Take Control dialog box



3. Select a joystick configuration from the Configuration list. (If you have specified a joystick configuration as a favorite, you do not have a choice of the configuration to use.)
4. Select the entity functions you want to control (movement). You can have more than one joystick attached and can also use keyboard controls, if they are configured.
5. Click OK.

81.1.1 Switching Between Systems

Systems that are controlled by joysticks can be organized into functional groups so that you can configure them similarly and switch between the systems using a "switch between" key or button.

You can create your own switch groups. For details, see "81.3.4 Configuring “Switch Between” Options" on page 1585.

➤ **To rename a system function group:**

1. On the Start menu, choose **MAK Technologies > VR-Forces 5.2 Simulation Object Editor**. The Simulation Object Editor opens.
2. Load *EntityLevel.sms* or your own entity-level SMS.
3. Select the entity whose system you want to edit.
4. Select the Movement tab.
5. .
6. Click the Edit Properties button ( ) for the Movement Dynamics system. A Properties dialog box opens.
7. Edit the group name. For example, change `Automotive Driving` to `Tracked Vehicle Driving`.
8. Click OK.
9. Save the SMS.

For complete details about using the Simulation Object Editor, see "Chapter 68. The Simulation Object Editor and SMSs" on page 1303 through "Chapter 73. Configuring Embarkation" on page 1441.

81.1.2 Disabling Joystick Control

➤ **To disable joystick control:**

1. Select the entity for which you want to disable joysticks.
2. Choose **Objects > Joystick > Release Control**.

81.2. Using the Keyboard to Control an Entity

The joystick configuration process allows you to configure keyboard control of entity movement and weapons. You can use keyboard control along with joysticks.

To enable or disable keyboard control, follow the procedures for enabling and disabling joystick control. Similarly, to configure keyboard control, follow the joystick configuration process. For more information, see "81.1. Using a Joystick to Control Entities" on the previous page and "81.3. Configuring Joysticks and Keyboard Control" below.

81.3. Configuring Joysticks and Keyboard Control

You can control entity movement and weapon fire using a joystick, the keyboard, or both. You can attach multiple joysticks and can use a mix of joystick and keyboard control at the same time. You configure joystick and keyboard mappings on the Application Settings dialog box, Joystick Configuration page.

The controls available for configuration on the Joystick Configuration page are taken from the joystick controllers in the movement, gun, and other system definitions. For example, the *ground-tracked.sysdef* file has the following entry:

```
(joystick
  (component-descriptor-type "joy-controller-descriptor")
  (component-type  "ground-move-joystick-controller")
   ...
  (joystick-controls
    (steering
      (function-name "steering")
      (function-group $automotive-driving)
      (description "Controls the steering of this entity")
      (min-value -1.000000)
      (max-value 1.000000)
    )
    (throttle
      (function-name "throttle")
      (function-group $automotive-driving)
      (description "Controls the throttle of this entity")
      (min-value 0.000000)
      (max-value 1.000000)
    )
    (brake
      (function-name "brake")
      (function-group $automotive-driving)
      (description "Controls the braking of this entity")
      (min-value 1.000000)
      (max-value 1.000000)
    )
    (change-gear
      (function-name "change-gear")
      (function-group $automotive-driving)
      (description "Puts the entity in forward, reverse or
          pivot gear")
      (min-value 0.000000)
      (max-value 2.000000)
      (cycle True) ;; Each click cycles from min to max
    )
  )
)
```

Based on this system definition file, the Joystick Configuration page (Figure 641 on the facing page) has an Automotive Driving function with controls for steering, throttle, brake, and change-gear.

The **min-value** and **max-value** parameters determine how the edit buttons on the Joystick Configuration page work:

- If **min-value** and **max-value** are both set to 1 (as in the brake control in the example system definition file), the control is a key mapping button (⌨). You can set it to Yes or No.

- If **max-value** is an integer greater than 1 (as in the change gear control in the example system definition file), the control is a key mapping button. You can configure keyboard buttons to increase or decrease the control by 1, or to set it to a specific value.
- If **min-value** does not equal **max-value** and **max-value** is less than or equal to 1 (as in the throttle control in the example system definition file), there is a joystick button () and a key mapping button on the Joystick Configuration page. The keyboard button lets you specify a percentage increase or decrease of the control value. (For example, increase throttle by 10%.)

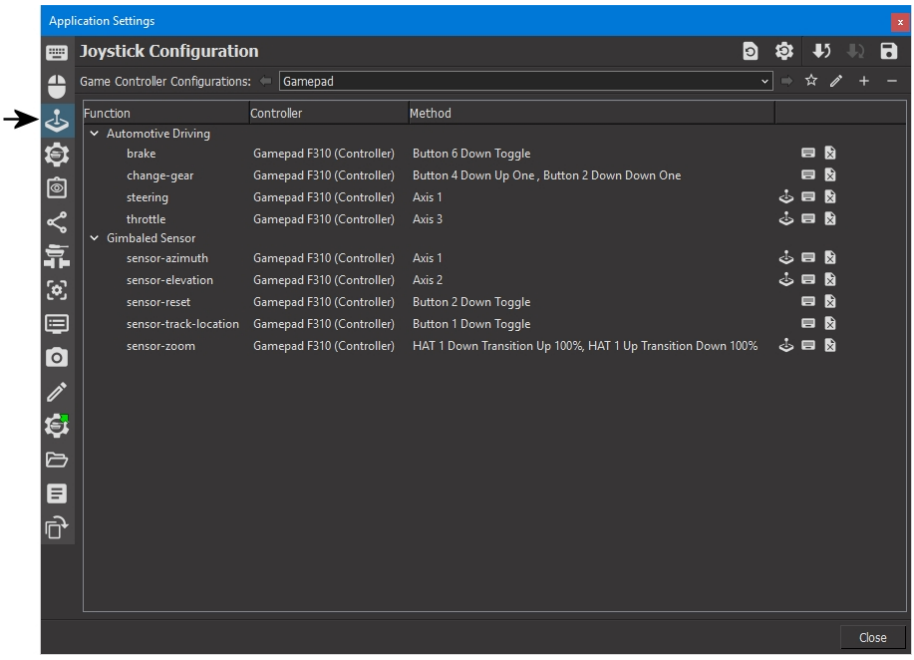
81.3.1 Creating a Joystick Configuration

A joystick configuration contains the joystick mappings and, optionally, keyboard mappings that you can use to control an entity. VR-Forces includes default configurations for a game controller and a joystick.

➤ **To create a joystick configuration:**

1. Create or open a scenario.
2. Choose **Settings > Application**. The Application Settings dialog box opens.
3. Select the Joystick Configuration page (Figure 641).

Figure 641. Joystick Configuration page

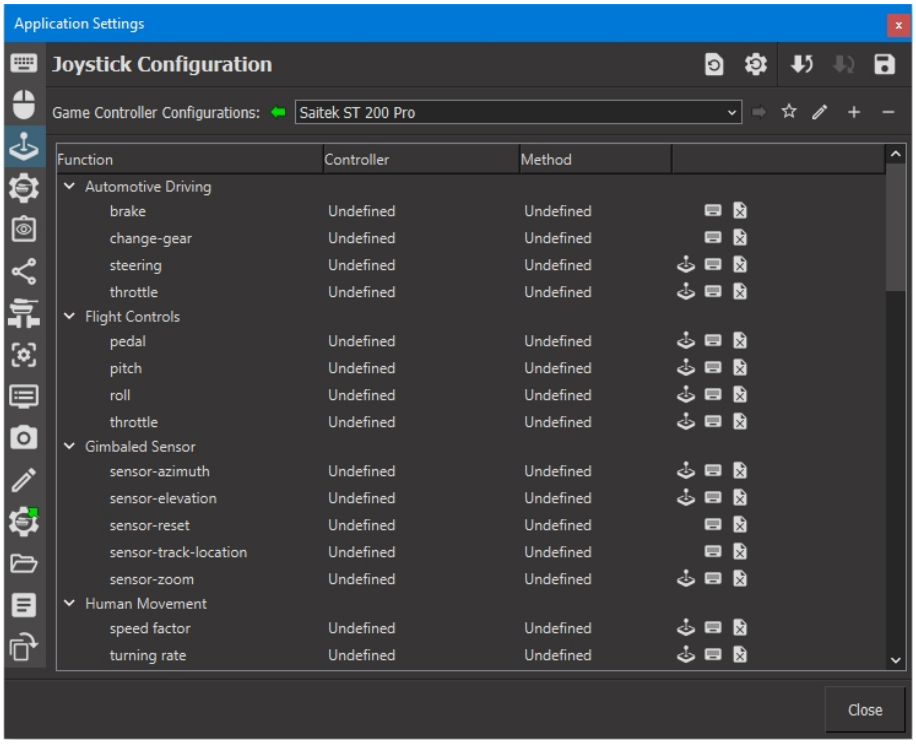


4. Click the Add button (). The New Configuration dialog box opens.
5. Enter a name for the configuration.

> ⚠ The name of the joystick configuration must match the name that the joystick uses to identify itself to the computer.

6. Click OK. The window redisplays to show the functions that this device can control (Figure 642).

Figure 642. Joystick configuration



7. Edit the joystick and keyboard mappings as described in "81.3.2 Mapping the Joystick Stick" below and "81.3.3 Mapping Joystick Buttons and Keyboard Keys" on the facing page.

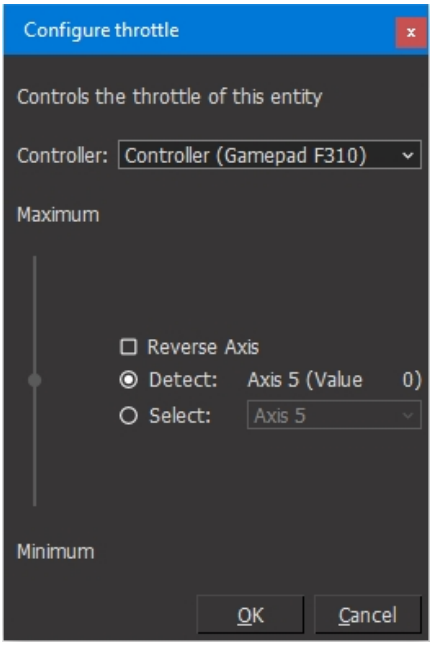
81.3.2 Mapping the Joystick Stick

To control an entity with a joystick, you map the joystick controls to specific entity movement components and weapons.

➤ **To map joystick controls:**

1. Create a joystick configuration, as described in "81.3.1 Creating a Joystick Configuration" on the previous page .
2. On the Joystick Configuration page (Figure 641 on the previous page), click the Joystick button (🕹) for the function you want to map. A joystick configuration dialog box opens (Figure 643).

Figure 643. Joystick configuration dialog box



3. In the Controller list, select the controller you want to configure.
4. Configure each axis:
    a. To let VR-Forces select the axis, select the Detect option and move the joystick in the axis you want to use for this control.
    b. To choose the axis, select the Select option and select an Axis from the list.
    c. Move the joystick in the axis you are configuring. If the Maximum/Minimum slider moves in the direction that you expect, you are finished. If the slider moves opposite to what you expect, select the Reverse Axis check box. It should now move as expected.
5. Click OK.
6. Repeat this process for each function you want to control with the joystick.

81.3.3 Mapping Joystick Buttons and Keyboard Keys

You can map joystick buttons and keyboard keys to entity movement control and weapon control. When entity control is enabled, the entity control key mappings override any other key mappings, such as observer control mappings.

➤ **To map joystick buttons and keyboard keys to entity control:**

1. Create a joystick configuration, as described in "81.3.1 Creating a Joystick Configuration" on page 1581.
2. On the Joystick Configuration page (Figure 641 on page 1581), click the Key Mapping button ( ) for the control you want to map. A configuration dialog box opens. It may call for a range of values (Figure 644), or just be a toggle (Figure

645).

Figure 644. Key mapping configuration dialog box

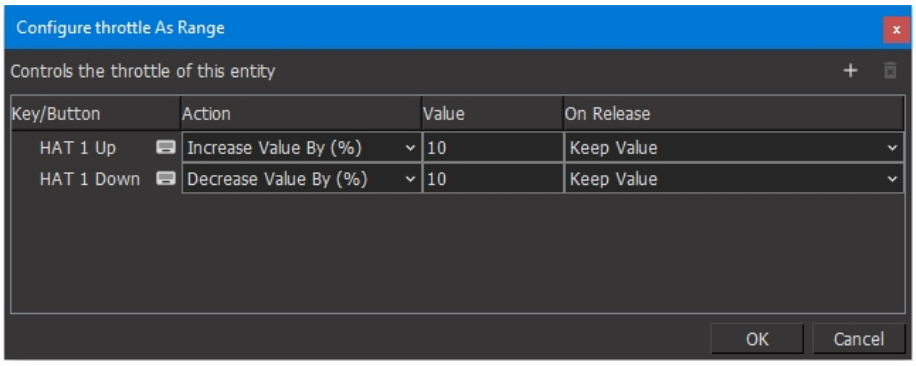
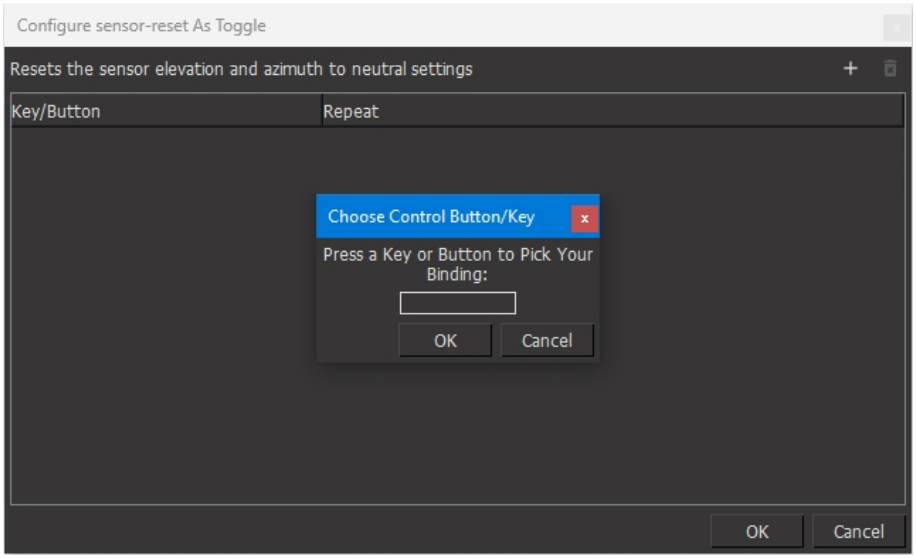
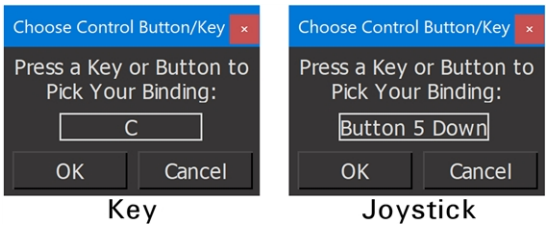


Figure 645. Key mapping configuration as toggle dialog box



3. If the mapping requires a range, for each row in the dialog box:
    a. Click the Key Mapping button ( ). The Choose Control Button/Key dialog box opens (Figure 646).
    b. Press the joystick button or keyboard key that you want to use to activate this control. It displays in the dialog box (Figure 646). (If a key binding is not available, the OK button is not enabled.)

Figure 646. Choose Control Button/Key dialog box with selection



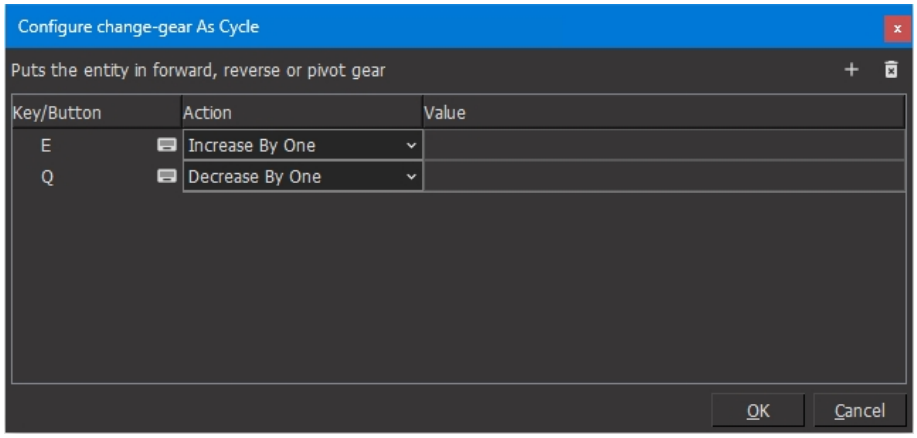
c. Click OK.

d. In the Action column, select an option from the list.

e. In the Value column, enter an appropriate value for the Action option.

f. Specify other settings as appropriate for the mapping.

Figure 647. Key mapping configuration dialog box



4. If the keyboard mapping is a toggle, the Choose Control Button/Key dialog box opens automatically. Do the following:

   a. Press the key or joystick button that you want to use.

   b. Click OK.

   c. In the Repeat column, select Yes or No. Yes means that the action repeats if you hold down the key. No means it happens one time for each key press.

5. Click OK.

6. Repeat this process for each function you want to configure.

81.3.4 Configuring “Switch Between” Options

Some entities may have multiple systems of a certain type, such as weapons. You might want to be able to control all similar systems with the same set of keys and joystick controls. VR-Forces lets you do this with switch groups. You can specify a key or button to press to switch between the different systems in a switch group. For example, you might press a button to switch between a tank's main battle gun and a machine gun.

The controls for which you can specify a switch between option are listed in the Switch Between section of the Joystick Configuration page. Specifying the key or button to use is described in "81.3.3 Mapping Joystick Buttons and Keyboard Keys" on page 1583. This section explains how to set up switch options.

To set up a switch option, when you define a function group in a system definition file's metadata, you use the syntax `switch_option:system`. The `switch_option` is a keyword. If an entity has multiple systems that use that keyword, you can switch between them using the switch between key or button. VR-Forces systems use the **weapon** keyword for switching between weapon systems. The following code from *mm-gun.template_sysdef* shows how this works.

The controllers specify two function groups for joystick control: `$slew-group` and `$ballistic-gun-group`. These groups specify four functions that the joystick can control: slew, elevation, fire, and change ammo:

```
(weapon-mm-gun-template
 ...
  (controllers
    (turret-joystick
     ...
      (joystick-controls
        (slew
          (function-name "slew")
          (function-group $slew-group)
          (description "Controls the slew of this weapon")
          (min-value -1.000000)
          (max-value 1.000000)
        )
      )
    )
    ...
    (main-gun-control-joy
     ...
      (joystick-controls
        (elevation
          (function-name "elevation")
          (function-group $ballistic-gun-group)
          (description "Controls the elevation of this weapon")
          (min-value -1.000000)
          (max-value 1.000000)
        )
        (fire
          (function-name "fire")
          (function-group $ballistic-gun-group)
          (description "Fires this weapon")
          (min-value 1.000000)
          (max-value 1.000000)
        )
        ...
        (change-ammo
          (function-name "change ammo")
```

```
            (function-group $ballistic-gun-group)
            (description "Changes the ammo of this weapon")
            (min-value 1.000000)
            (max-value 1.000000)
          )
        )
```

The meta-data section of the sysdef file specifies a value for these groups. It is specified as the keyword weapon followed by the name of the system – weapon:120mm Ballistic Gun:

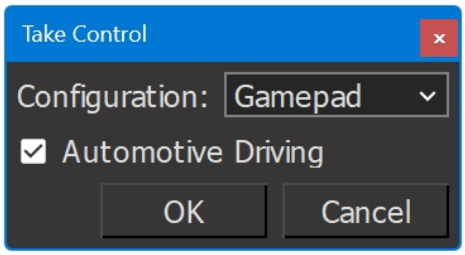
```
  (meta-data
   (system-name "Ballistic Gun (Elevating/Rotating)")
   (system-description "Template for a system that will allow for the
Simulation Object to be equipped with a rotating, elevating mounted
ballistic weapon.")
   (allowed-state-repository-types "ground-vehicle-param"
      "surface-entity-param" "rotary-wing-entity-param")
   (system-categories "weapon")
   (parameter-data-list
      ...)
      (string-parameter-data
         (parameter-name  "slew-group")
         (variable-type  "DtRwString")
         (display-label  "Slew Joystick Group Name")
         (display-units  "")
         (source-units  "")
         (list-items "")
         (list-display 0)
         (allow-edit True)
         (display-value True)
         (roll-up-rule "none")
         (default-value $default-slew-group
      )
      (string-parameter-data
         (parameter-name  "ballistic-gun-group")
         (variable-type  "DtRwString")
         (display-label  "Ballistic Gun Joystick Group Name")
         (display-units  "")
         (source-units  "")
         (list-items "")
         (list-display 0)
         (allow-edit True)
         (display-value True)
         (roll-up-rule "none")
         (default-value $default-ballistic-gun-group)
      )
  ...
```

Other weapon systems are configured similarly.

81.3.5 Using Multiple Joysticks

You can attach multiple joysticks or game controllers to your computer. You must create a configuration for each joystick that you want to use. When you take control of an entity, you select the configuration you want to use and the functions that you want to control with the selected configuration.

Figure 648. Take Control dialog box



You could select one configuration to control both, or you could control the two functions with different controllers.

➤ **To control two functions with different controllers:**

1. Select the entity.
2. Choose **Objects > Joystick > Take Control**. The Take Control dialog box opens.
3. Choose the controller configuration.
4. Select the check boxes for the features you want to control with the selected configuration and clear the others.
5. Click OK.
6. With the entity still selected, choose **Objects > Joystick > Take Control** again. The check boxes that are available are those not controlled by the previous configuration.
7. Click OK.

81.3.6 Specifying a Favorite Joystick Configuration

Ordinarily, when you ask to take control of an entity with a joystick, you must specify the joystick configuration to use. You can specify a configuration as the favorite. In that case the configuration is automatically applied. However, you cannot choose a different configuration.

➤ **To specify a favorite joystick configuration:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Joystick Configuration page (Figure 641 on page 1581).
3. In the Game Controller Configurations list, select the configuration that you want to specify as the favorite.
4. Click the Star button (☆). A star is added to the configuration name in the list.

81.4. Installing a Gamepad on Linux

You can use Logitech Gamepads on Linux. You configure them like you would a joystick.

➤ **To install a Gamepad driver on Linux:**

1. Download the xboxdrv driver for Linux.
2. Install the driver:

   ```
   yum install xboxdrv
   ```
3. Test to see if your controller is recognized:

   ```
   xboxdrv -L
   ```

   Your device should be listed win an ID next to it.
4. To test that the device is working:

   ```
   xboxdrv -i id
   ```
5. To have the device work with VR-Forces, issue the command:

   ```
   xboxdrv --detach-kernel-driver --silent
   ```



82. Attaching Components to Remote Entities

This chapter explains how to configure components on remote entities.

82.1. Attaching Components to Remote Entities 1592

82.1. Attaching Components to Remote Entities

You can attach VR-Forces sensors, actuators, and controllers to remote (non-VR-Forces) entities. This feature allows you to add to the capabilities of the remote entity and better integrate it with the VR-Forces simulation. For example, you could add a radio to an entity or you could enable spot reporting. When you attach a component to a remote entity, you can write a plan for that entity that uses the attached components. For details, see "43.10. Writing Plans for Remote Entities" on page 980.

VR-Forces includes an example scenario, remoteAttachment, that uses remote attachment. Its use requires that you run an external application, the f18 example that comes with VR-Link. For details about how to run the scenario, please see the scenario's description. This and other example scenarios are found in *./userData/scenarios/Sample*.

When you add a component to a remote entity, the application that is simulating the entity is responsible for updating its position and health status. VR-Forces simulates the attached components. You can specify that VR-Forces simulate the components on one sim engine or across all sim engines.

Simulation of components on remote entities has limitations:

- Only individual platforms and lifeforms can attach components. Units and environmental process objects are not supported.
- VR-Forces components cannot control remote entities. That is, they cannot change their position or state. Therefore, you should not attach components such as movement systems to remote entities, because they will have no effect on the entities. Similarly, components that expect to be attached to a locally simulated entity, such as an articulated part, may not work properly when attached to remote entities.
- By default, remote entities are not added to the echelon hierarchy. The result is that they do not send or receive spot reports. If you want spot reporting for remote entities, set `connection-mode` to `force` in the *commModelParams.mtl* for the appropriate SMS.

Components are attached to remote entities on the basis of entries in a component attachment table. The component attachment table provided by VR-Forces is *./data/simulationModelSets/<model_set>/extra/componentAttachmentTable.mtl*.

The component attachment table associates entities with attachment key-names. The following is a sample component attachment table:

```
(remote-configurations
 (attachment-entry
    ;; CIS MIG 29
    (attach-by object-type (obj-type 1 (1 2 222 1 2 1 -1)))
    (key-name "mig29-config")
 )
 (attachment-entry
    ;; US F-18
    (attach-by object-type (obj-type 1 (1 2 225 1 9 -1 -1)))
    (key-name "f18-config")
 )
```

```
)
```

The entries in this table associate the key-names mig29-config and f18-config with appropriate entity type enumerations. You can also associate a key-name with marking text:

```
(attachment-entry
 (attach-by marking-text "HMMWV 1")
 (key-name "hmmwv-attachments")
)
```

The key-names in the component attachment table have corresponding entries in *vrfSim.opd* in the remote-configurations block:

```
(remote-configurations
 (remote-configuration
    (key-name "f18-config")
    (configuration-path "$(remote-config-dir)/f18.mtl")
 )
 (remote-configuration
    (key-name "mig29-config")
    (configuration-path "$(remote-config-dir)/mig29.mtl")
 )
)
```

The key-names are associated with specific configuration files, which by default are located in *./data/simulationModelSets/<model_set>/vrfSim/remoteConfigurations*. The configuration files contain the components that are to be attached to the remote entity. For examples, please see the files in the *./data/simulationModelSets /EntityLevel/vrfSim/remoteConfigurations* directory.



XIV. Configuring the Graphical User Interface

The chapters in this section describe the display engine, windows, and channels, explain features of the graphical user interface (GUI), and how to configure menus and toolbars.



83. Display Engine and Window Management

This chapter explains how to add and configure windows and channels and how to save display engine configurations.

83.1. The Display Engine 1598
83.1.1 Window Types 1598
83.2. Managing Display Engine Configurations 1598
83.2.1 Adding a Window 1598
83.2.2 Adding a Channel to a Window 1601
83.2.3 Removing a Window 1602
83.2.4 Removing a Channel 1602
83.2.5 Saving a Display Engine Configuration 1602
83.2.6 Loading a Display Engine Configuration 1602
83.3. Changing a Window's Attributes 1603
83.4. Changing a Channel's Attributes 1604
83.4.1 Setting the Clipping Planes 1605
83.4.2 Specifying the Projection Resize Policy Attribute 1607
83.4.3 Changing a Channel's Frustum (Field of View) 1609
83.4.4 Changing the Viewport 1610
83.4.5 Configuring Water Visibility 1611
83.4.6 Configuring a Mirror Channel 1612
83.4.7 Channel Keywords 1613
83.5. Configuring Multichannel Displays 1613
83.5.1 Changing the Camera's Position and Orientation Offset 1615
83.6. Stereoscopic Displays 1616
83.6.1 Configuring Anaglyphic Stereo 1617
83.6.2 Configuring Polarized Stereo 1617

83.1. The Display Engine

The VR-Forces GUI has a component called a display engine. The display engine renders 3D and 2D data and can display data in one or more windows, which can each have zero or more channels. The windows and channels being rendered at any particular time make up a display engine configuration. You can save display engine configurations and can load them to quickly recreate a particular configuration.

83.1.1 Window Types

VR-Forces supports the following types of windows:

- Resizable (Window). A resizable window can be moved, resized, maximized, and minimized. It does not have any GUI controls. An inset view is an example of a resizable window. (For information about inset views, see "62.1. Inset Views" on page 1224.)
- Full screen. A full screen window uses the entire area of a monitor. It does not have any menus, panels, or other GUI controls.
- Embedded. An embedded window is part of an application. The main display window in the VR-Forces GUI is an embedded window. The application provides a variety of controls that affect the view in the window. You could, if you wanted to, change VR-Forces's window to a different type, in which case it would no longer be embedded in the main application window.
- Frameless Window. A single window that can span multiple screens without a frame around it. This improves the framerate performance because it is one window. Useful for a flight station that uses multiple monitors.

A window can have zero or more channels. Each channel defines a viewable area on the screen. You must have at least one channel to see a rendered image in the window. Each channel can have its own observer.

83.2. Managing Display Engine Configurations

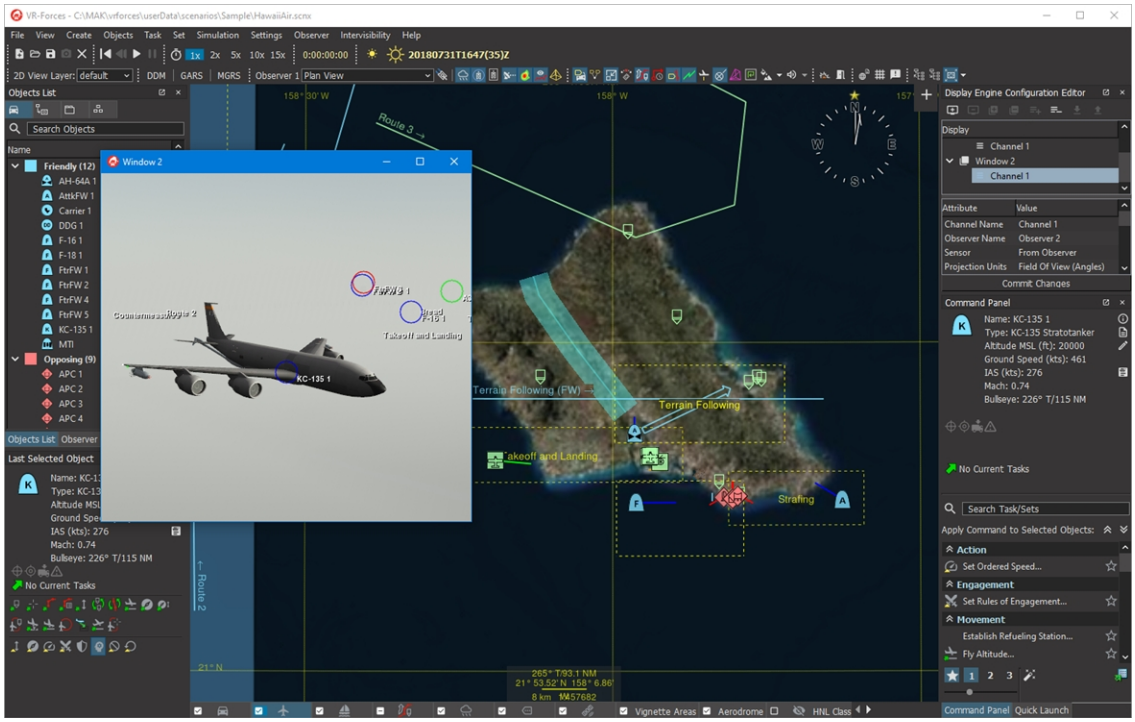
You can add windows to the display engine and you can add channels to the windows. The windows and channels make up a display engine configuration. You can save a display engine configuration and load a saved configuration.

83.2.1 Adding a Window

You can add as many windows as you want. New windows are named for the window type, for example Stealth Inset 1, Stealth Inset 2, Plan View Inset 1, and so on. Each window is created with one channel. You can add new windows from the File menu or in the Display Engine Configuration Editor.

➤ To add a window from the File menu, choose **File > New Stealth Window** (or **New Plan View Window**). A new window opens (Figure 649).

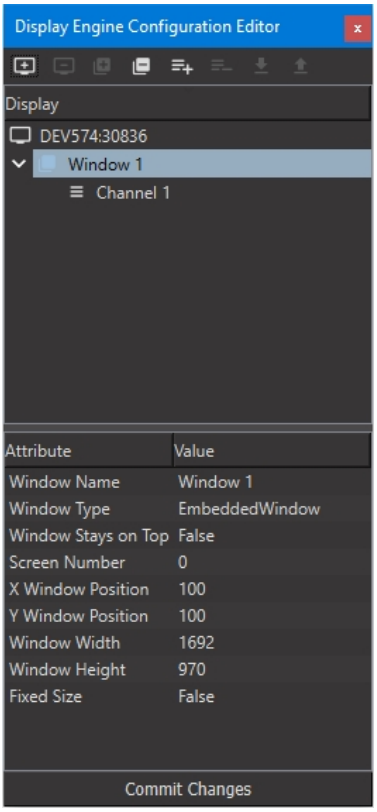
Figure 649. New window



➤ **To add a window from the Display Engine Configuration Editor:**

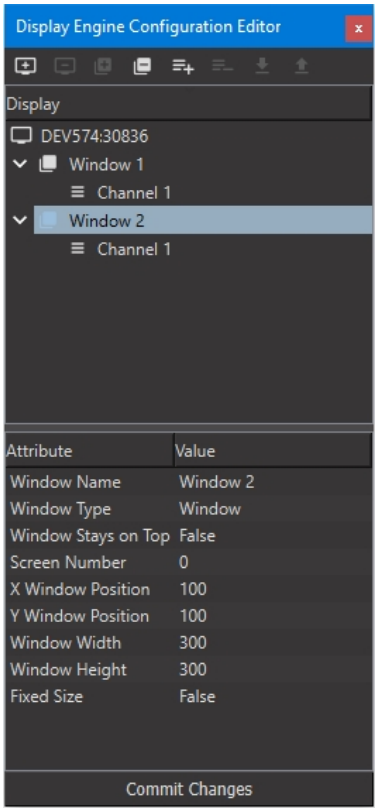
1. Choose **View > Display Engine Configuration Editor Panel**. The Display Engine Configuration Editor Panel is added to the VR-Forces window (Figure 650).

Figure 650. Display Engine Configuration Editor Panel



2. Select the display engine at the top of the Display tree.
3. Click the Add a Window button (), or right-click the display engine name and choose **Add a Window** on the context-sensitive menu. A new window opens and is listed in the Display Engine Configuration Editor Panel (Figure 651).

Figure 651. New window



83.2.2 Adding a Channel to a Window

Adding a channel to a window lets you configure multiple views within that window.

➤ **To add a channel to a window:**

1. In the Display Engine Configuration Editor Panel, select the window to which you want to add a channel.
2. Click the Add a Channel button (), or right-click the window name and choose **Add a Channel** from the menu.

To see a difference between the channels, change the attributes of the new channel. For details about editing channel attributes, see "83.4. Changing a Channel's Attributes" on page 1604.

83.2.3 Removing a Window

➤ **To remove a window:**

1. In the Display Engine Configuration Editor Panel, select the window that you want to remove.
2. Click the Remove Window button (), or right-click the window name and choose **Remove Window** from the menu.

83.2.4 Removing a Channel

➤ **To remove a channel:**

1. In the Display Engine Configuration Editor Panel, select the channel that you want to remove.
2. Click the Remove Channel button (), or right-click the window name and choose **Remove Channel** from the menu.

83.2.5 Saving a Display Engine Configuration

You can save a display engine configuration so that you can easily replicate the configuration at a later time.

➤ **To save a display engine configuration:**

1. Click the Save Display Engine Configuration button (), or right-click the name of the display engine and then choose **Save Display Engine Configuration** on the menu. The Save Display Engine Configuration dialog box opens.
2. Type a name for the display engine configuration.
3. Click Save.

83.2.6 Loading a Display Engine Configuration

➤ **To load a display engine configuration:**

1. In the Display Engine Configuration Editor Panel, select the display engine at the top of the window.
2. Click the Load Display Engine Configuration button (), or right-click the name of the display engine and choose **Load Display Engine Configuration** on the menu. The Load Display Engine Configuration dialog box opens.
3. Select the display engine configuration you want to load.
4. Click Open.

> You can also drag-and-drop a *.DCX* file to the Display Engine Configuration Editor Panel to load it.

83.3. Changing a Window's Attributes

You can change the attributes of a window:

- Window Name.
- Window Type (for details about window types, see "83.1.1 Window Types" on page 1598).
- Window Stays on Top.
- Screen Number.
- Position (X and Y coordinates of upper left corner, in screen pixels).
- Width and height, in pixels.
- Fixed size.

➤ **To change a window's attributes:**

1. Select the window in the Display Engine Configuration Editor Panel. A list of attributes displays at the bottom of the editor (Figure 650 on page 1600).
2. Click the value of the attribute that you want to change.
3. To change:
   - A numeric value, type or select a value.
   - The Window Type or Fixed Size, select a value from the list.
   - The name, type a new name.
   - Whether the window should always display on top of other windows, set Window Stays on Top to True.
4. Click Commit Changes. The window updates.

You can also change a window's position by dragging it to a new location. You can change its size by resizing the window directly. The new values display in the attributes list the next time you select the window in the Display Engine Configuration Editor Panel.

83.4. Changing a Channel's Attributes

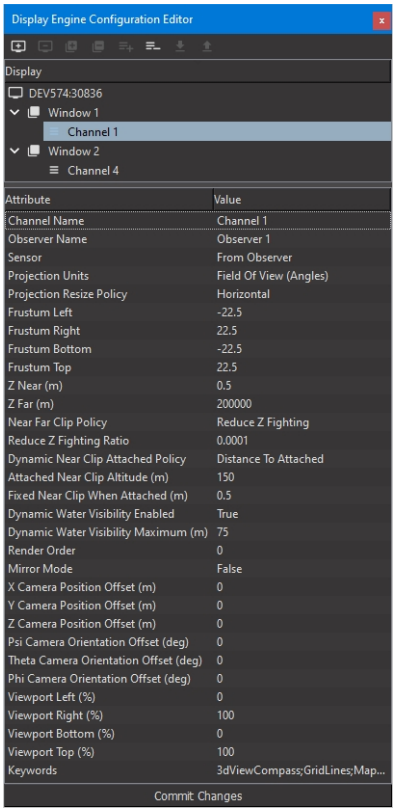
You can change attributes of a channel:

- Channel Name.
- Observer Name.
- Sensor. Use the sensor configured for the current observer, or override the observer and use one of the listed sensors.
- Projection Units – field of view, in angles, or database, in meters.
- Projection Resize Policy – fixed, vertical, or horizontal.
- Frustum (Left, Right, Bottom, and Top) (field of view).
- Z Near and Z Far (clipping planes).
- Near Far Clip Policy.
- Reduce Z Fighting Ratio.
- Dynamic Near Clip Attached Policy.
- Attached Near Clip Altitude.
- Fixed Near Clip When Attached.
- Dynamic Water Visibility Enabled.
- Dynamic Water Visibility Maximum.
- Render Order. Set the order in which channels are rendered in a particular window. For example, to display Channel 1 over Channel 2, set Channel 1 to render after Channel 2.
- Mirror Mode. Flips the channel horizontally.
- Camera Position Offset (X, Y, Z).
- Camera Orientation Offset (Psi (heading), Theta (pitch), Phi (roll)).
- Viewport (Left, Right, Bottom, Top).
- Keywords – keywords to associate with the channel, for example, "showCockpits". (For a list of channel keywords, see "83.4.7 Channel Keywords" on page 1613.)

➤ **To change a channel's attributes:**

1. Choose **View > Display Engine Configuration Editor Panel**.
2. Select the channel in the Display Engine Configuration Editor Panel. A list of attributes displays at the bottom of the editor (Figure 652).

Figure 652. Channel attributes



3. Click the value of the attribute that you want to change.
4. To change:
   - A numeric value, type or select a new value.
   - The Observer Name, Sensor, Project Units, Projection Resize Policy, or Parent Channel, select a value from the list.
   - The Channel Name, type a new name. (For details about some of these attributes, see the sections that follow this one.)
5. Click Commit Changes. The window channel updates.

83.4.1 Setting the Clipping Planes

To improve performance, the graphics engine does not render anything that falls outside of a specified range of distances from the observer. This range is determined by the near and far clipping planes. Clipping planes are set on a per-channel basis.

Figure 653 illustrates the effect of changing the near clipping plane (the Z Near attribute). Figure 654 illustrates the effect of changing the far clipping plane (the Z Far attribute). To see the effect of changing the near and far clipping planes, you must set the Near Far Clip Policy to Fixed.

Figure 653. Near clipping

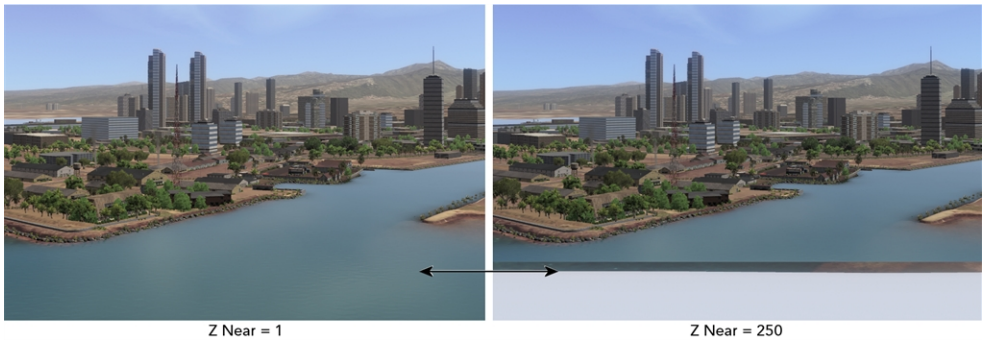
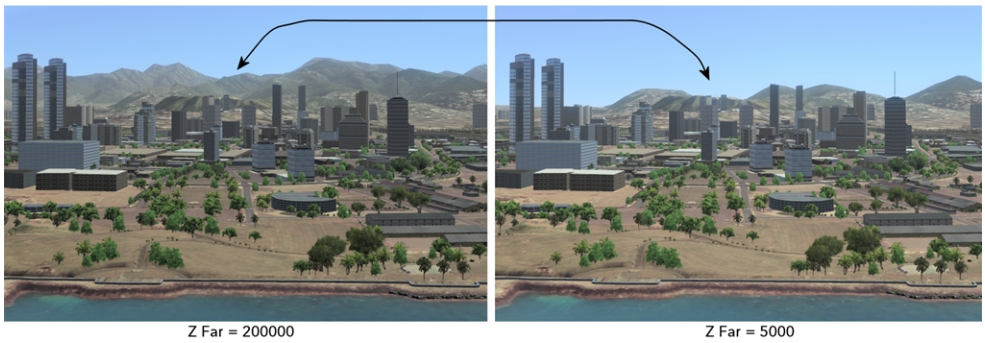


Figure 654. Far clipping



The ratio between the far clipping plane and the near clipping plane determines the precision with which the renderer can determine differences in depth. If this ratio becomes too small, then terrain and objects that are distant may not sort correctly, resulting in what is referred to as Z-fighting. VR-Forces provides different policies for managing the way near and far clipping planes are determined.

VR-Forces sets the clipping planes using one of these policies (the Near Far Clip Policy attribute), plus their modifiers:

- Fixed. Clipping planes are set explicitly using the Z Near and Z Far attributes.
- Reduce Near Clip. Clipping planes are adjusted dynamically based on the observer's altitude. The near clipping plane is increased slightly as the observer rises to 20,000 meters, but stays under 1.5 meters. When the observer rises above 20,000 meters, the near and far clipping planes gradually increase. The Reduce Near Clip policy can be modified by the Dynamic Near Clip Attached Policy.
- Reduce Z Fighting. Clipping planes are set based on a ratio of the near clipping plane to the far clipping plane (the Reduce Z Fighting Ratio attribute). The near clipping plane can be as large as needed to maintain the ratio. This policy is the default clipping plane policy. The Reduce Z Fighting policy can be modified by the Dynamic Near Clip Attached Policy.
- Clamp Near Clip When Magnified. It is the same as Reduce Z Fighting, but adds an intersector to clamp the near clipping plane to prevent the near clip from going past objects or terrain in the scene when using view magnification. The intersector is only used when view magnification is 2x or more (this value can be changed

through the API by extending the channel and overriding the call that calculates the clamp). This is an expensive operation, so only enable it if you really need it.

The Reduce Z Fighting clipping plane policy greatly reduces Z-fighting when you are using dynamic ocean. However, if the observer is attached to a simulation object and its altitude is moderately high, it may be closer to the entity than the near clipping plane and the entity may be clipped away. You can adjust the ratio to provide the best results. If you are not using dynamic ocean or you are frequently attaching to entities, you may want to use one of the other clipping plane policies.

If you set the Near Far Clip Policy to Reduce Z Fighting or Reduce Near Clip and the observer is attached to a simulation object, VR-Forces can dynamically change the near clipping plane based on the Dynamic Near Clip Attached Policy:

- Ignore Attached. Do not take entity attachment into account when setting the near clipping plane.
- Fixed Near Clip. Set the near clipping plane to the value of the Fixed Near Clip When Attached attribute if the observer altitude is >= the value of the Attached Near Clip Altitude attribute.
- Distance To Attached. Use the distance between the observer and the attached entity to determine the near clipping plane if the observer altitude is >= the value of the Attached Near Clip Altitude attribute. When this policy is selected, VR-Forces compares the near clipping value without this modifier to the value with it and uses the smaller of the two values. This is the default policy.



- Reverse Z buffering is enabled by default to reduce Z fighting. It is not supported in 2D observer modes. The Oculus Rift HMD does not work with reverse Z buffering. To disable reverse Z buffering, start VR-Forces with the `--disableReverseZBuffer` command-line option.
- When using Reverse Z Buffering in conjunction with Reduce Z Fighting Near Far Clip Policy (the default) and Triton dynamic ocean, significant performance slowdowns sometimes occur. Changing the Near Far Clip Policy to Fixed resolves the issue; when Reverse Z Buffering is enabled, the Reduce Z Fighting policy is not needed.

83.4.2 Specifying the Projection Resize Policy Attribute

The Projection Resize Policy determines how a channel behaves when you resize it. The options are:

- Fixed. Keep the aspect ratio the same regardless of how the window is resized.
- Horizontal. Change the horizontal field of view (FOV) to so that the ratio of the FOV angles matches the aspect ratio of the viewport.
- Vertical. Change the vertical FOV so that the ratio of the FOV angles matches the aspect ratio of the viewport.

You can change the projection resize policy for individual channels by changing the Channel Projection Resize Policy attribute. You can also change the projection resize policy for all channels. Figure 655 illustrates the effects of the different resize policies: (1) Horizontal Resize Policy, (2) Fixed Resize Policy, and (3) Vertical Resize Policy. (The observer remains at the same position.)

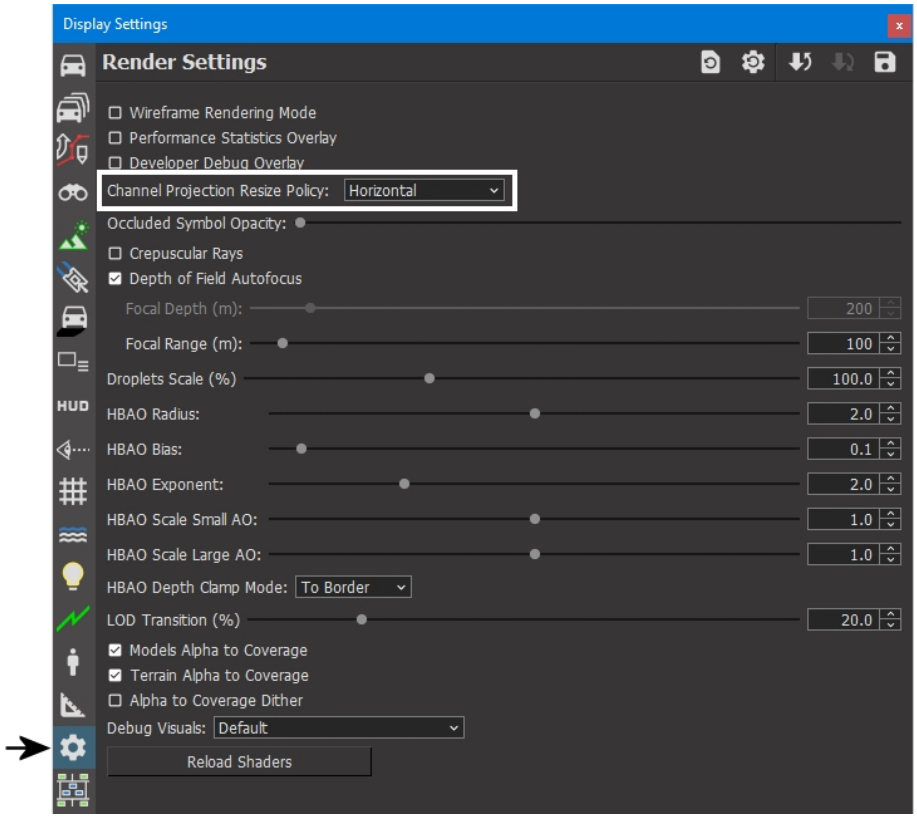
Figure 655. Effect of different projection resize policies



➤ **To change the projection resize policy for all channels:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Render Settings page (Figure 656).

Figure 656. Display Settings, Render Settings page



3. Select an option on the Channel Projection Resize Policy list.

83.4.3 Changing a Channel's Frustum (Field of View)

The field of view setting affects perspective in a manner similar to a camera lens.

A wide field of view creates an effect like that of a wide-angle lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth distances between objects become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed.

Figure 657 illustrates different field of view settings taken from the same observer location. The view on the left has frustum values -22.5, 22.5, -22.5, 22.5. The view on the right has the frustum values -10, 10, -10, 10.

Figure 657. Field of view

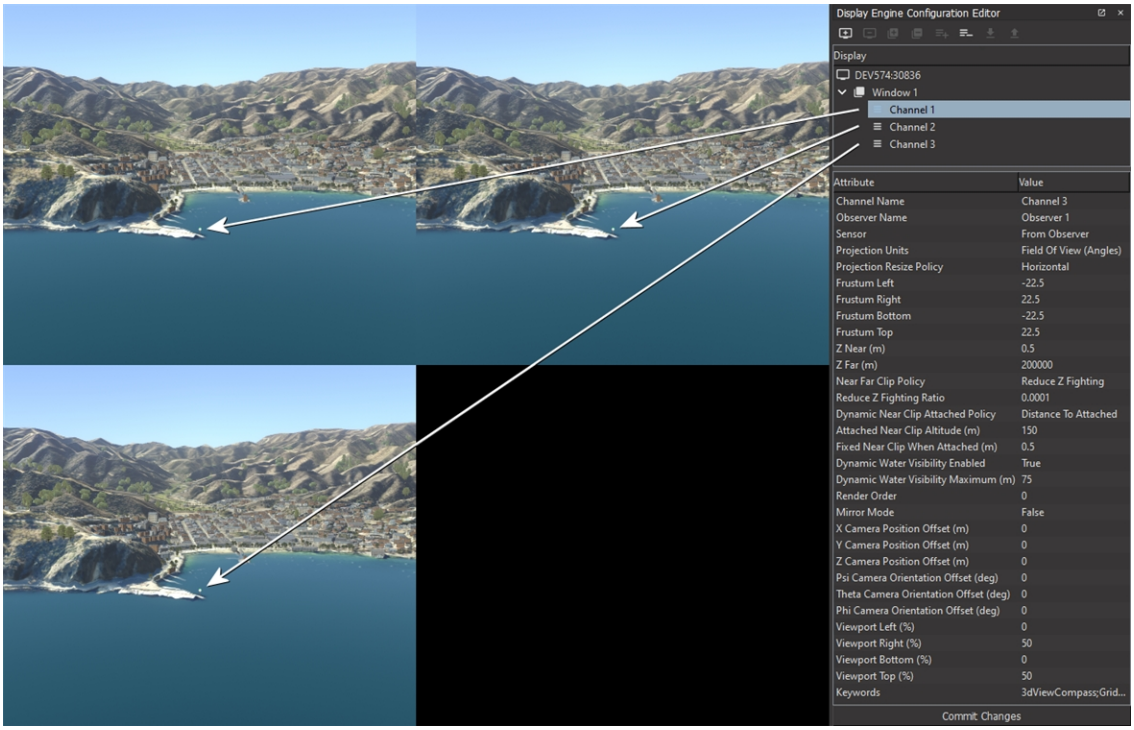


➤ To change a channel's field of view, edit the frustum values in its attributes list.

83.4.4 Changing the Viewport

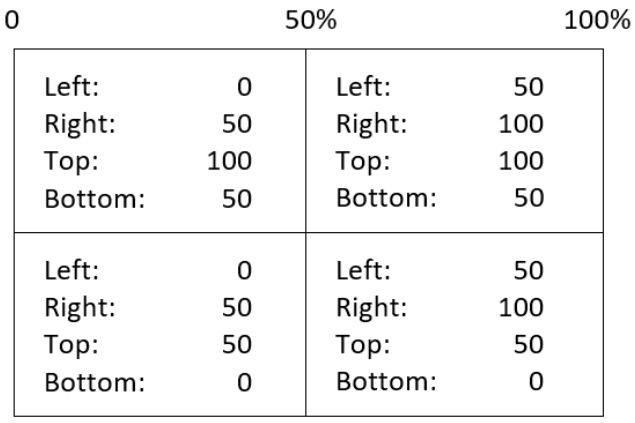
When you change the viewport of a channel, you change the portion of the window in which the channel displays. This does not change the extents of the scene. A narrow viewport compresses the scene width. A short viewport compresses the height. Figure 658 shows a window that has three channels, each with a different viewport. (The arrows show which portion of the window corresponds to each channel.) Notice that each channel has the same field of view.

Figure 658. Window with three channels and three viewports



Viewports are specified as a percentage of the window from left to right and top to bottom. Figure 659 shows the percentage allocation for the viewports in Figure 658 on the previous page.

Figure 659. Viewport configuration



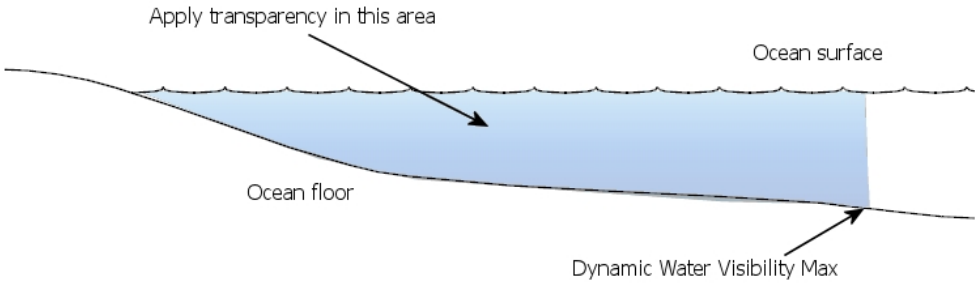
- When you change viewports, the window might not completely refresh. To see the changes, resize the window.
- If you have viewports that are adjacent, such as when setting up a FOV of more than the default (such as 90 or 270 degrees instead of 45), you must disable HDR and HBAO+. See "A.5. Seamless Scenes with Multiple Channels" on page 1661.

83.4.5 Configuring Water Visibility

The water visibility attributes (along with the surface transparency setting on the Environment Conditions dialog box, Weather page) are designed to reduce Z-fighting. Z-fighting occurs when the altitude of the ocean surface and the depth of the ocean floor are very close to each other, usually around coastlines that have bathymetry data. The likelihood of Z-fighting decreases as the difference between the ocean surface and ocean floor increases, but it increases as the observer's altitude increases (because the ocean surface and ocean floor are proportionally closer to each other). Therefore there is a dynamic relationship as these factors change.

Adding transparency to the water reduces Z-fighting. The Dynamic Water Visibility Enabled attribute increases transparency automatically as the observer's altitude increases. Dynamic Water Visibility Maximum specifies the maximum depth to which transparency is added (Figure 660).

Figure 660. Dynamic water visibility



Dynamic water visibility is enabled by default. When Dynamic Water Visibility Enabled is True, VR-Forces increases the water visibility depth from the value Surface Transparency setting (at 0 meters altitude), to the maximum value specified at the current Ocean LOD Elevation. The default depth is 75 meters. This value provides reasonable performance given the default Ocean LOD Elevation (5000 meters), the altitude above which VR-Forces stops displaying dynamic ocean. If you change the Ocean LOD Elevation so that dynamic ocean displays when the observer is higher than 5000 meters, Z-fighting may increase and you may need to increase the Dynamic Water Visibility Maximum value. (For details about setting the ocean LOD Elevation, see "Setting the Ocean LOD Elevation" in the *Adding Content to MAK ONE Applications Reference Manual*.)

There is no performance cost to enabling this attribute when you are not using dynamic ocean. If there are no underwater terrain polygons, these attributes have no effect.

83.4.6 Configuring a Mirror Channel

You can enable the Mirror Mode for a channel to view it as a horizontal mirror of the scene. This is useful in situations where you want to include a view such as a rearview mirror. This procedure describes how to do this when VR-Forces is connected to a scenario that includes a ground vehicle.

➤ **To set up a channel as a mirror:**

1. In either the terrain or the Objects List, right-click on the vehicle and then choose an observer attach mode, for example, **Observer Attach > Attach Mimic**.
2. Choose **View > Display Engine Configuration Editor Panel**.
3. Select the window and then click Add a Channel ( ). The channel is added to the list under the window.
4. Modify the properties of your new channel:
    a. Change the Viewport settings to position the channel near the top of the window. For example: Left 20, Right 80, Bottom 75, and Top 90.
    b. Set the Observer Psi to 180. This points the camera behind the viewer.
    c. Set Mirror Mode to True.
    d. If there is a cockpit that you do not want to include, remove showCockpits from the Keywords list.
    e. Click Commit Changes.

Figure 661. Observer attached to a truck and displaying a rearview mirror



83.4.7 Channel Keywords

You can edit channel keywords in the Keywords attribute of each channel in the Display Engine Configuration Editor. VR-Forces supports these channel keywords:

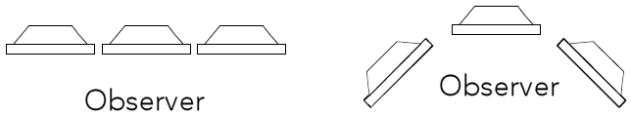
- largeCompass. If the compass is enabled, this channel shows the large compass. Default: Included in configuration of all channels.
- smallCompass. If the compass is enabled, this channel shows the small compass. Default: Not included in the configuration of any channels.
- GridLines. If grid lines are enabled, this channel shows the grid lines. Default: Included in configuration of all channels.
- showRemoteGraphics. If remote graphics are received this channel displays them. Default: Included for all channels except inset views.
- showCockpits. If GL Studio cockpits are enabled, this channel shows them. Default: Included in configuration of all channels.
- NoExternalInput. This keyword disables all mouse/keyboard/touch events for the channel. Default: Only included in the configuration of Sensor Gimbal View channels.
- cockpitOnly. If this keyword is true, the scene will not be rendered in the channel. Default: Included in the configuration of instrument panels only.

83.5. Configuring Multichannel Displays

People running simulations often want to set up a multichannel configuration using multiple monitors (on one or more computers) to display exercises. Typical configurations are a horizontal alignment, which widens the view, and the angled alignment, which surrounds the observer to simulate an out-the-window view (Figure 662).

You could set up these display configurations by having one computer with several monitors attached, or you could have multiple computers, each running a VR-Forces GUI. You might also want to have a 2D view on one monitor and a 3D view on another.

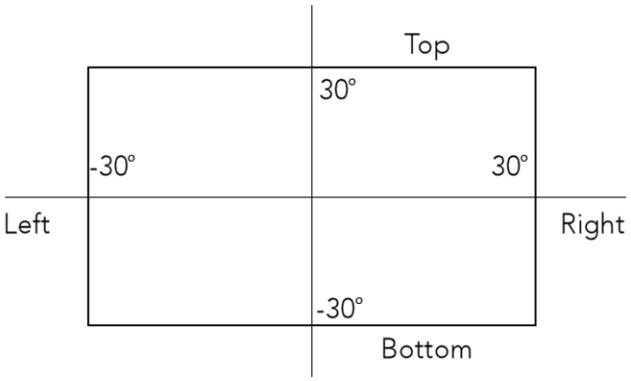
Figure 662. Multi-channel displays (top-down view)



By default, all windows and channels show the same view of the terrain (Figure 664, default window views). Therefore, if you want a setup in which each channel shows a different view of the terrain, you must configure the channels to calculate an offset from the master view.

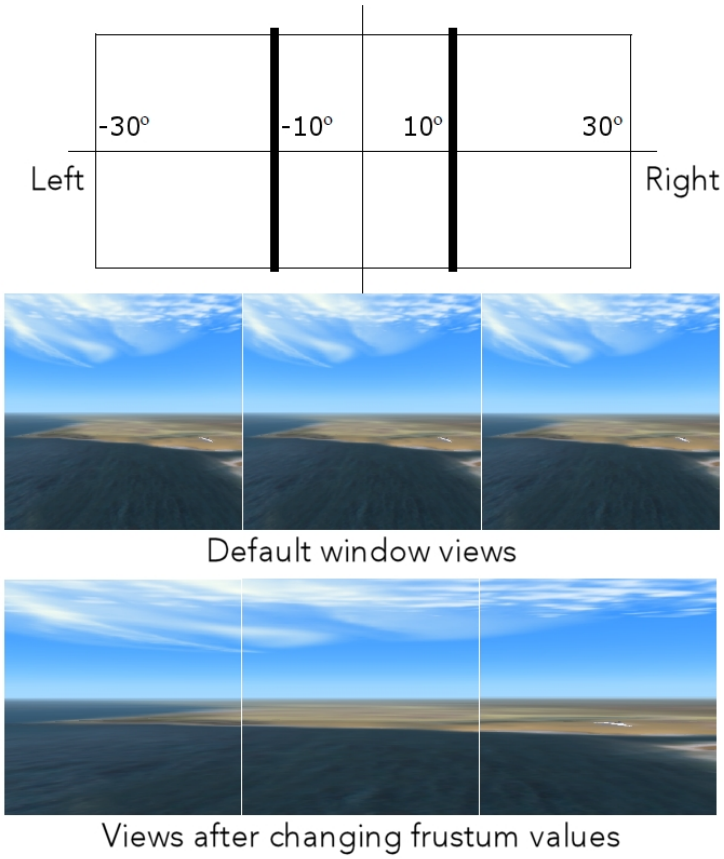
The distribution of the view across multiple channels is controlled by the frustum. Figure 663 illustrates the default frustum for Channel 1. It has a 60º field of view, from -30º to +30º.

Figure 663. Default frustum for Channel 1



If you wanted to split the view into, for example, three horizontal channels, you would edit the left and right frustum values for each channel, as shown in Figure 664. The field of view is the same: -30º to +30º. However, now each channel shows a 20º segment of the total field of view.

Figure 664. Frustum values for three channels



Once you configure the windows and channels you need to create the views you want, save the display engine configuration on each computer that you have configured. The next time you run the scenario, load the appropriate display engine configurations on each computer.

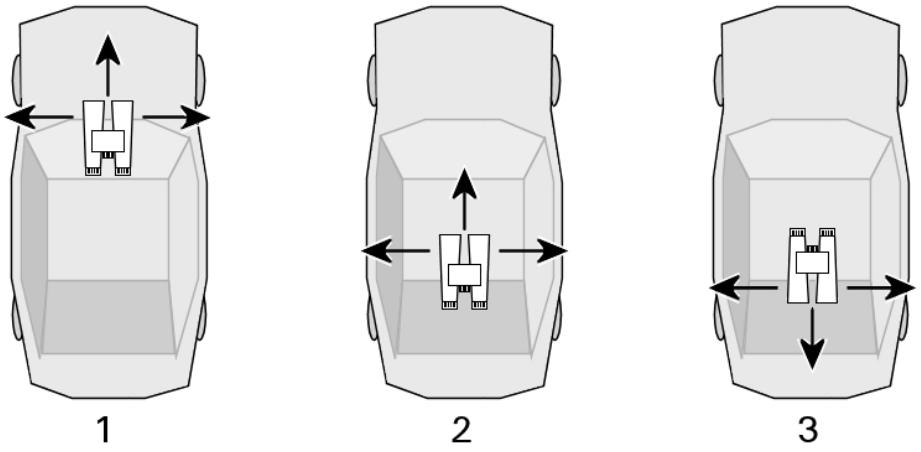
For a multi-monitor configuration to work correctly, use fixed size windows whose size matches the aspect ratio specified by the horizontal and vertical fields of view and set the projection resize policy to fixed. (For information about the projection resize policy, see "83.4.2 Specifying the Projection Resize Policy Attribute" on page 1607.)

83.5.1 Changing the Camera's Position and Orientation Offset

Changing the frustum affects how much of the database is in the view. Changing the camera's position and orientation affects the point from which you look at the view. For example, imagine you are simulating the view from a vehicle, as in vehicle 1 in Figure 665. (The camera location is represented by the binoculars.)

If you wanted to show the view as experienced by a gunner riding in the back of the vehicle, or in a turret, you would change the X, Y, and Z camera position offsets in the channel's attribute list (vehicle 2).

Figure 665. Position and orientation offsets



If you wanted the gunner to look out the rear of the vehicle, you would change its orientation with the Psi (heading), Theta (pitch), and Phi (roll) Camera Orientation Offsets (vehicle 3).

83.6. Stereoscopic Displays

Stereoscopic display produces realistic three dimensional imagery by showing each of the viewer's eyes a different image, rendered from that eye's point of view. A stereoscopic display requires a system that can render two channels, one for each eye, and a method of displaying the correct channel to the correct eye of the viewer. Three common display methods are active stereo, polarized display, and anaglyphic stereo. OpenSceneGraph can support all three of these methods, and since VR-Forces uses OpenSceneGraph for most of its rendering, all three should be possible in VR-Forces.

- Active Stereo: In an active stereo setup, frames are alternated on the display for the left eye and the right eye, and a pair of shutter glasses blocks light to either the left or right eye in synchronization with the display. This method requires hardware support to synchronize the glasses to the display, along with a graphics card capable of quad buffering, and a display that can refresh at twice the desired frame rate. Active stereo should be possible with VR-Forces, but has not been tested.
- Anaglyphic Stereo: Anaglyphic stereo is the cheapest and simplest method to use. A typical setup is a display showing a red channel and a cyan channel simultaneously, along with a pair of glasses with one red lens and one cyan lens for the viewer. This will work with no special hardware aside from the inexpensive glasses. It is also the simplest method to set up in VR-Forces, requiring only that some OpenSceneGraph environment variables be set.
- Polarized Stereo: A polarized display shows the frames for both eyes simultaneously, but with their light polarized in different directions. The viewer wears glasses that have lenses polarized to match the two different frames, allowing each eye to see only the frame intended for it. Polarized stereoscopy requires some specialized equipment to display the images with the correct polarization. One simple method for producing this kind of display is to use two projectors with polarizing filters, projecting onto a rear projection screen. Monitors are available

that can polarize the two frames. It is easy to configure a polarized display with VR-Forces.

83.6.1 Configuring Anaglyphic Stereo

You can configure OpenSceneGraph for anaglyphic stereo rendering through environment variables. To use VR-Forces in anaglyphic mode, run it with the environment variables described in Table 74.

Table 74. Environment variables for anaglyphic stereo

Environment Variable	Description	Default
OSG_STEREO	Turn stereo on.	ON
OSG_STEREO_MODE	Use anaglyphic stereo when in stereo.	ANAGLYPHIC
OSG_SCREEN_DISTANCE	Specifies the distance, in meters, that the viewer is from the screen.	0.50
OSG_SCREEN_HEIGHT	Specifies the height of the image, in meters, on the screen.	0.26
OSG_SCREEN_WIDTH	Specifies the width of the image, in meters, on the screen.	0.325
OSG_EYE_SEPARATION	Specifies the eye separation (interoccular distance).	0.06

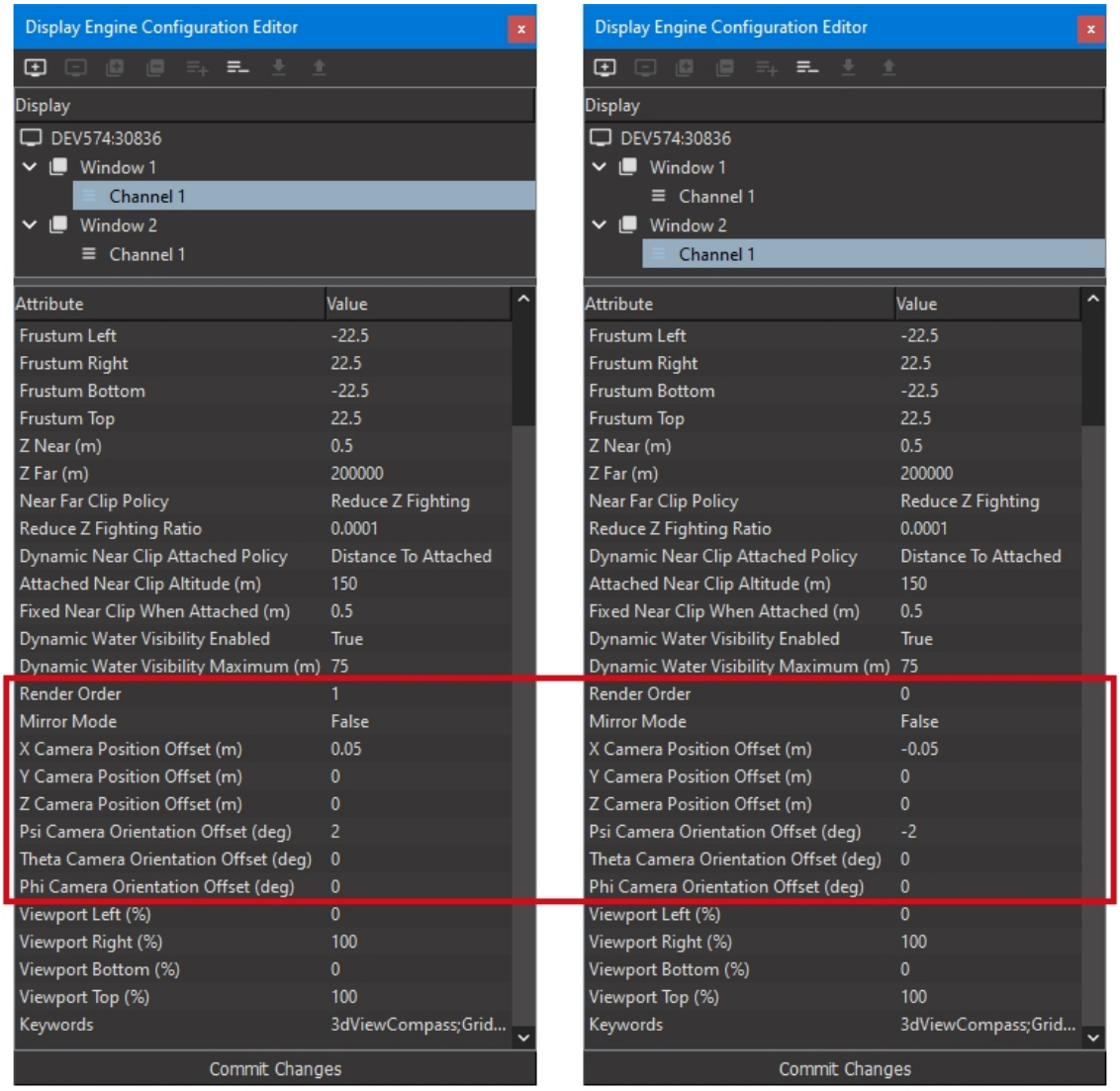
For more information, please see the OpenSceneGraph documentation at http://www.openscenegraph.org/index.php/documentation/guides/user-guides/62-stereo-settings.

83.6.2 Configuring Polarized Stereo

You can configure polarized stereo by taking advantage of VR-Forces's ability to produce multichannel displays. You configure one channel per eye. Each channel renders with a camera that is at a slight offset from the other channel's camera to represent the spacing between the viewer's eyes.

Varying the eye spacing and focal length can produce different effects, and the best values may depend on what the viewer is looking at. Figure 666 shows an example configuration.

Figure 666. Polarized stereo channel configurations



➤ **To compute the correct channel configuration values:**

1. Pick a focal length, L. Objects at this distance will appear to be in the plane of the display. Objects that are further away will appear behind the display, and objects closer will appear in front of it.
2. Pick an eye spacing, E, to represent the distance between the viewer's eyes. Human eyes are typically about 0.063m apart[1].

[1]"Variation and extrema of human interpupillary distance", N. A. Dodgson, Proc SPIE 5291. Select publications link and scroll to 2004 at http://www.cl.cam.ac.uk/~nad10/pubs/EI5291A-05.pdf.

3. Set the X Camera Position values to `–E/2` meters for the left channel and `E/2` meters for the right channel.
4. Set the Psi Camera Orientation values to `–arctan(E/2)/L` degrees for the left channel and `arctan(E/2)/L` degrees for the right channel.



84. GUI Controls, Layouts, and Behaviors

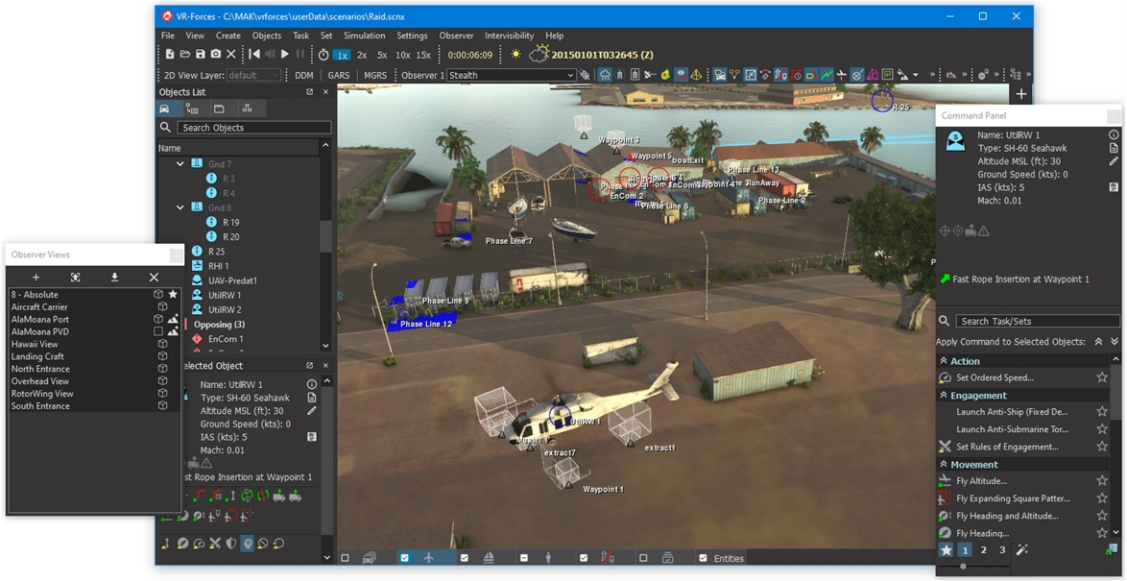
This chapter provides details about using the various panels and toolbars in the VR-Forces GUI.

84.1. Dockable Panels and Toolbars 1622
84.1.1 Docking and Undocking a Panel 1622
84.1.2 Docking Panels as Tabs 1622
84.1.3 Docking and Undocking a Toolbar 1623
84.1.4 Docking a Toolbar Vertically 1623
84.1.5 Displaying and Hiding Panels and Toolbars 1624
84.2. Saving Window Layouts 1624
84.2.1 Creating a Window Layout 1624
84.2.2 Choosing the Window Layout 1625
84.2.3 Setting a Window Layout as the Default Layout 1626
84.2.4 Reverting a Window Layout 1627
84.2.5 Updating a Window Layout 1627
84.2.6 Renaming a Window Layout 1627
84.2.7 Deleting a Window Layout 1628
84.2.8 Restoring a Factory Layout 1628
84.3. Using Tear-Off Menus 1628
84.4. Viewing the Window in Full-Screen Mode 1629
84.4.1 Hiding the Mouse in Full Screen Mode 1629
84.5. Using Context-Sensitive Menus 1630
84.6. Common VR-Forces Buttons 1630
84.7. Using the Text Chat 1631
84.7.1 Configuring Text Chat Channels for a Scenario 1632
84.8. Adding Text to the Title Bar 1633
84.9. Specifying Display Units 1634
84.10. Disabling and Enabling Dark Mode 1637

84.1. Dockable Panels and Toolbars

VR-Forces has floating, dockable panels for many of its important functions (Figure 667). It also has toolbars that let you quickly access the commands available on the menus.

Figure 667. VR-Forces window and undocked panels



84.1.1 Docking and Undocking a Panel

➤ **To undock a panel:**

1. Place the cursor over the panel's title.
2. Press and hold the left mouse button.
3. Drag the panel away from the window.

ⓘ You can also undock a panel by clicking the pop out button ⧉.

➤ To redock a floating panel where it came from, double-click its title bar.

84.1.2 Docking Panels as Tabs

You can have multiple panels in the same part of the GUI by making them tabbed panels.

➤ **To dock a panel as a tab:**

1. Place the cursor over the panel's title.
2. Press and hold the left mouse button.
3. Drag the panel on top of another panel so that it is highlighted.

84.1.3 Docking and Undocking a Toolbar

➤ **To undock a toolbar:**

1. Place the cursor over the handle.

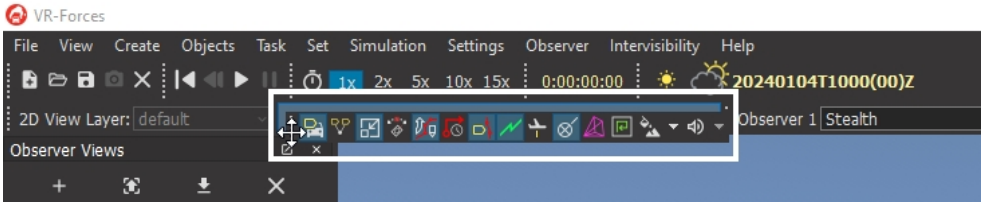
Figure 668. Toolbar handle



2. Press and hold the left mouse button.
3. Drag the toolbar away from the window.

➤ To dock a toolbar, drag it back to the toolbar strip. The other toolbars move to create a space to drop in the toolbar (Figure 669).

Figure 669. Docking a toolbar

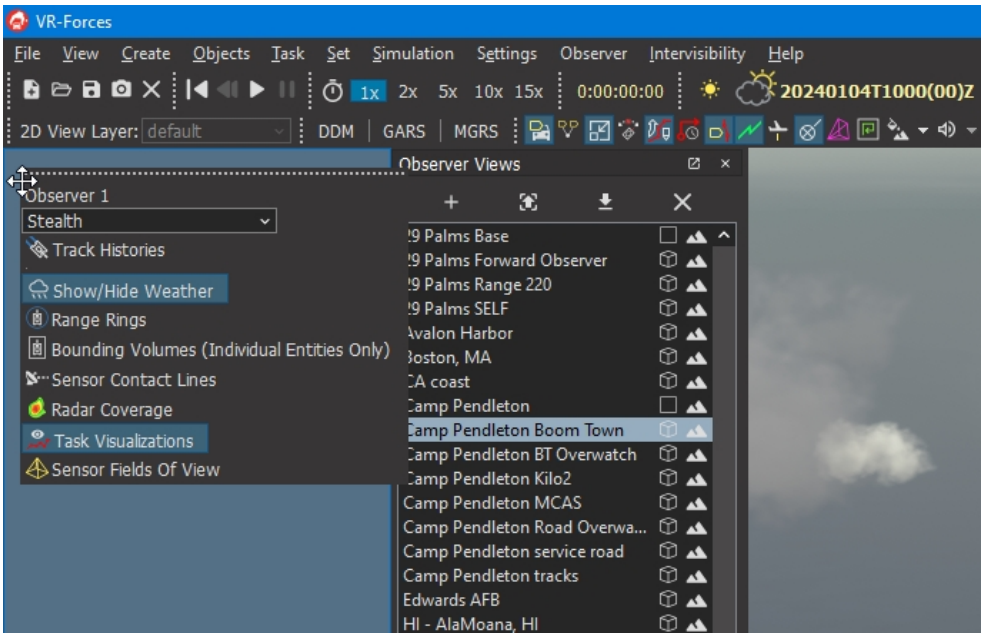


84.1.4 Docking a Toolbar Vertically

You can dock a toolbar vertically. Doing so displays the label for each button.

➤ To dock a toolbar vertically, drag it to the left- or right-hand edge of the window.

Figure 670. Vertically docking a toolbar



84.1.5 Displaying and Hiding Panels and Toolbars

You can hide any of the panels or toolbars.

➤ To hide or view a panel or toolbar, select it on the View menu.

84.2. Saving Window Layouts

You can save window layouts to named layouts and can easily switch between them while VR-Forces is running. A layout stores the window size and the visibility and location of all toolbars and panels. You can specify that a layout be the default layout used at startup.

Layouts are saved to *./appData/settings/vrfGui*. Each layout is saved to a separate file using the naming convention layout*_layout_name*.uisx. For backup, you can save layouts to *./factory/settings/vrfGui*.

You can see the available window layouts under **View > Window Layouts**. VR-Forces includes layouts:

- The Scenario Creation layout has a set of toolbars and panels that users typically need when creating a scenario.
- The Scenario Viewer layout removes the toolbars and panels to maximize the amount of the window that displays the scenario.
- There are ORBAT versions of these window layouts designed for using Order of Battle.

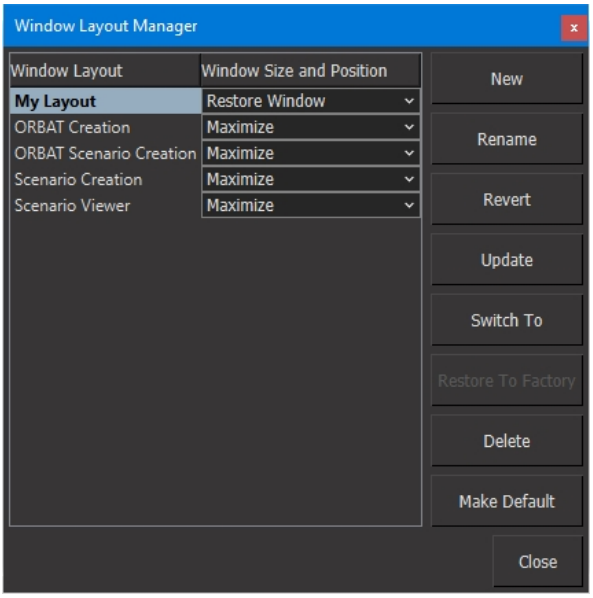
When you shut down VR-Forces, it saves the window layout. If you do not specify a default window layout, when VR-Forces starts up, it uses the saved window layout.

84.2.1 Creating a Window Layout

➤ **To create a window Layout:**

1. Arrange the toolbars, panels, and window size in the layout you want.
2. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671).

Figure 671. Layout Manager



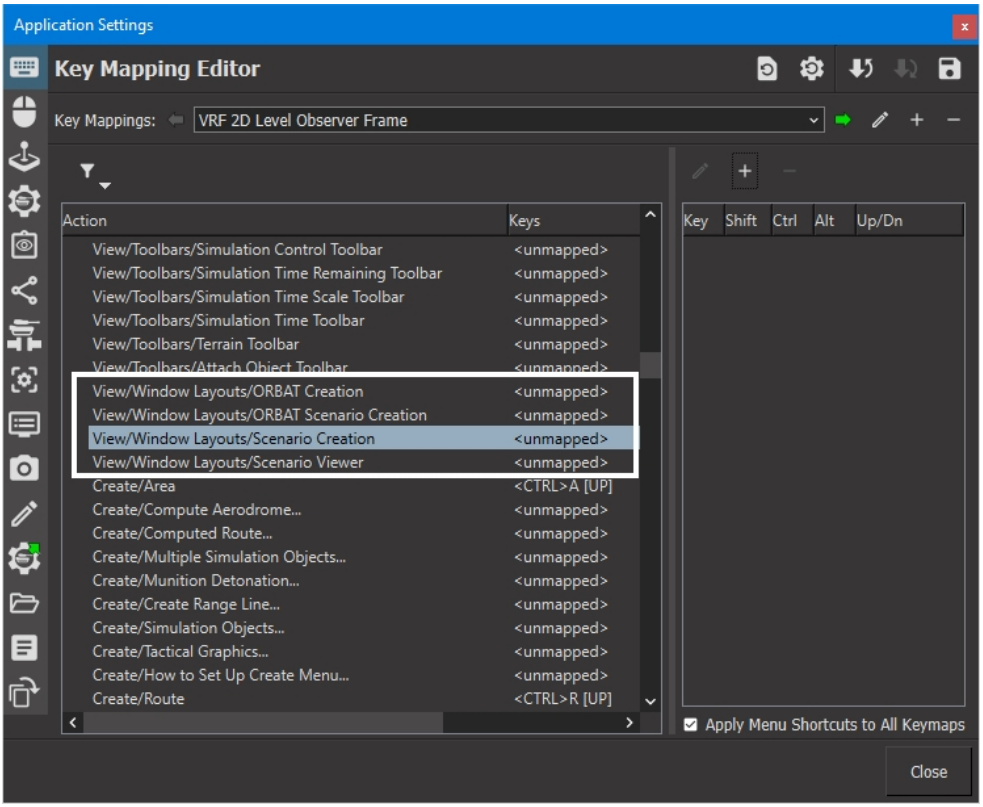
3. Click New. The Layout Name dialog box opens.
4. Type a name in the Layout Name box.
5. Click OK.
6. Select one of the window size options:
   - Maximize. Maximizes the window when this window layout is selected, regardless of the window size when the layout is created.
   - Restore Window. Save the current window size as part of the window layout. This is the default option.
   - Full Screen Mode. Put the window into full screen mode when this layout is selected, regardless of the window size when the layout is created.
7. Optionally, click Make Default to make your new layout (or whichever layout is selected) the layout you see at startup. A star appears next to the name of the default layout.
8. Click OK. The layout is added to the list of layouts.

84.2.2 Choosing the Window Layout

You can easily choose a window layout at any time. You can change layouts from the View menu or from the Layout Manager.

You can map keys to layouts to select them from the keyboard. Each layout can be mapped to a key (Figure 672). For details about mapping keys, see "Chapter 86. Creating and Editing Key Mappings" on page 1649.

Figure 672. Layout action in Key Mapping Editor



➤ To choose a window layout, choose **View > Window Layouts >** *layout*, where *layout* is one of the listed named layouts.

➤ To change the window layout in the Window Layout Manager, in the list of layouts, double-click the layout you want to use or select the layout you want and click Switch To.

84.2.3 Setting a Window Layout as the Default Layout

When VR-Forces starts, if there is a default layout, it uses that layout. Otherwise, it uses the window layout it saved the last time you shut down VR-Forces.

➤ **To set a window layout to be the default layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671 on the previous page).
2. In the list of layouts, select the layout you want to be the default.
3. Click Make Default. A star is displayed next to the name of the layout to indicate it is the default layout.

Clearing the Default Layout

If you have specified a default layout, you can specify that a different layout be the default or clear it so that there is no default layout.

➤ **To clear the default layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671 on page 1625).
2. To specify a different default layout, select the desired layout and click Make Default.
3. To just clear the current default layout, in the list of layouts, select the default layout and click Clear Default.

84.2.4 Reverting a Window Layout

If you use a named window layout and then rearrange the window, you can revert to the saved layout.

➤ To revert a window layout, choose **View > Revert Window Layout**. (You can also revert a layout on the Window Layout Manager dialog box.)

84.2.5 Updating a Window Layout

If you load a window layout and then make changes to the layout, such as resizing the window or closing a panel, VR-Forces does not keep track of the changes. If you switch to a different layout or shut down, the changes are lost. If you want to save changes to a layout, you must explicitly update it.

➤ **To update a window Layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager opens (Figure 671 on page 1625).
2. Select the layout you are using.
3. Click Update.

84.2.6 Renaming a Window Layout

You can change the name of a window layout.

➤ **To rename a window layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671 on page 1625).
2. Select the layout you want to rename.
3. Click Rename. The Layout Name dialog box opens.
4. Type a new name for the layout.
5. Click OK.

84.2.7 Deleting a Window Layout

You can delete window layouts that you have created.

➤ **To delete a window layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671 on page 1625).
2. Select the layout you want to delete.
3. Click Delete.

84.2.8 Restoring a Factory Layout

VR-Forces includes two layouts. They are in *./appData/settings/vrfGui*, which makes them available to the Layout Manager. They are also in *./factory/settings/vrfGui* so that they can be restored. All layouts created after you install VR-Forces are saved to *./appData/settings/vrfGui*. If you want to back up your layouts, you can copy them to any backup location you want. If you want to be able to restore a layout in the Layout Manager, copy it to *./factory/settings/vrfGui*.

To restore a layout from the factory, it must be present in *./appData/settings/vrfGui* so that you can select it in the Layout Manager. If you delete a layout and then want to recover it, if you have saved a copy of it to *./factory/settings/vrfGui* or anyplace else, you can copy it to *./appData/settings/vrfGui*.

➤ **To restore a window layout to the factory layout:**

1. Choose **View > Window Layout Manager**. The Window Layout Manager dialog box opens (Figure 671 on page 1625).
2. Select the layout you want to restore.
3. Click Restore From Factory.

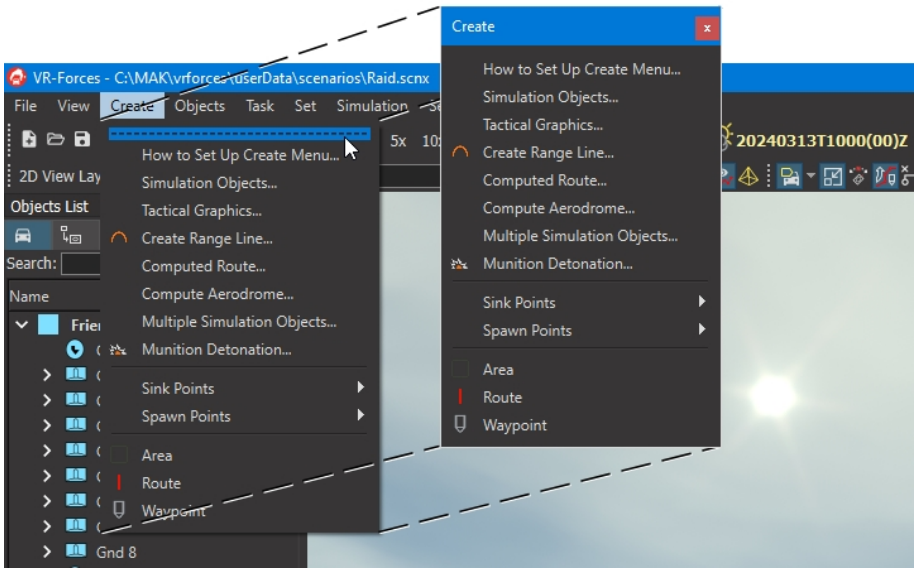
84.3. Using Tear-Off Menus

You can "tear-off" the Create menu and leave it displayed on the desktop.

➤ **To tear off the Create menu:**

1. Click the Create menu.
2. Move the cursor over the grab bar (dashed line) at the top of the menu to highlight it (Figure 673).
3. Click the grab bar. The menu is displayed as a window that you can leave open and reposition. (You can also continue to use the standard menu.)

Figure 673. Tearing off a menu



84.4. Viewing the Window in Full-Screen Mode

You can display VR-Forces so that it covers the entire screen and does not show the title bar or border. This mode is intended for use in demonstrations. When menus and panels are hidden, you can still use all keyboard controls, and you can still click simulation objects in the window to attach the observer to them.

You can configure VR-Forces to start in full screen mode with the `--fullScreen` command-line option. If you are viewing a simulation that takes place primarily over the ocean, you may see better performance using the `--pseudoFullScreen` command-line option.

➤ To view VR-Forces in full-screen mode, choose **View > Show Full Screen**, or press **Ctrl**+**Enter**.

➤ To exit full screen mode, press **Ctrl**+**Enter**.

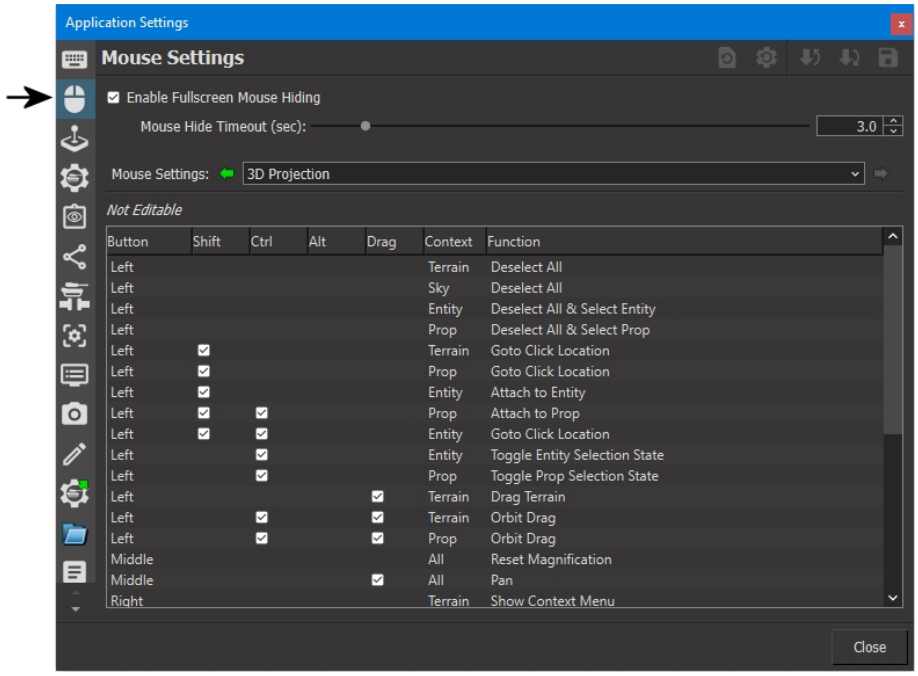
84.4.1 Hiding the Mouse in Full Screen Mode

Since full screen mode is often used only for display purposes, with no user interaction, the mouse is automatically hidden in this mode. As soon as you move the mouse, it displays. If you then stop using it, it is hidden when the threshold is reached.

➤ **To hide the mouse in full screen mode:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Mouse Settings page.

Figure 674. Mouse Settings page



3. Select the Enable Fullscreen Mouse Hiding check box.
4. Optionally, change the Mouse Hide Timeout value.

84.5. Using Context-Sensitive Menus

VR-Forces has context sensitive menus (also called shortcut menus, popup menus, or right-click menus) that you can display by clicking the right mouse button.

If you right-click a simulation object, tactical graphic, or other scene object, the menu displays the actions that you can take.

If you right-click the menu bar, you can select the commands available on the View menu.

84.6. Common VR-Forces Buttons

VR-Forces dialog boxes have a common set of buttons:

- Add an item (definition, model, and so on).
- Remove the selected item.
- Copy the selected item and add the copied version to the list.
- Select a filter of some type.
- Edit the selected item.
- Add from a list.
- Move up in the list.

- Move down in the list.
- Select the previous item.
- Select the next item.
- Add an attribute.
- Remove local attribute definition and use the parent's definition.
- Search.

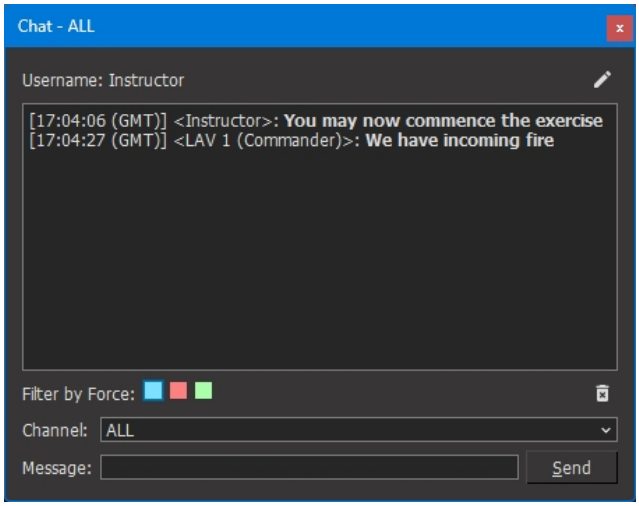
84.7. Using the Text Chat

The text chat feature was developed for use with VR-Engage, but you can also use it to chat between multiple VR-Forces GUIs connected to the same scenario. The timestamp shows the current time on the computer; it is unrelated to the simulation time.

- To change the name that displays in the chat window, use the Change Username button ( ). In Figure 675, the user changed their name to "Instructor". All users can see this name in their chat windows.
- To see the channels specific to one or more forces, select the force color next to Filter by Force. You can select multiple forces to access those channels.
- To see messages sent to a particular channel, choose a channel from the list. The default channels are All, which displays messages sent to any channel and is visible to all users, and channels for any selected forces, which include all users who are participating in that force. There might be channels defined for a scenario, which override the default channels as described in "84.7.1 Configuring Text Chat Channels for a Scenario" on the next page.
- To remove all messages from the currently selected channel, click the Clear Chat button ( ).

  Note that All messages are cleared from the window when you rewind a scenario.

Figure 675. Chat window



➤ To open a chat window, choose **View > New Chat Window**. You can open multiple windows if you want to have separate ones for different channels.

➤ To send a message, select a channel, type in the Message field, and then click Send.

84.7.1 Configuring Text Chat Channels for a Scenario

The default for channels in the chat window is to have one for each force and one for all participants. You can create a file for the scenario with channels specific to the scenario's purpose, such as channels for specific entities. For example, you might want channels for the members of each unit in the scenario, or for a particular task, such as a pilot and a JTAC.

- You must save a copy of the file in the *./userData/chatConfigs* folder for each installation that will use the channels. This is where VR-Forces and VR-Engage look when loading the scenario file. Additionally, the file name must be *scenario-file-name_ChatConfig.mtl*, where *scenario-file-name* is the exact name of the scenario file. The channels defined in those files are available to VR-Forces and VR-Engage users.
- You can configure channels that are specific to a force using its ID. The force IDs are defined in the Forces Editor, which is in the Simulation Object Editor. The default values are 0 for all forces, 1 for friendly, 2 for opposing, and 3 for neutral. Only users in a force can see messages sent to a channel configured for that force. If VR-Forces is configured to hide ground truth for a force (in Spot Report Settings, Figure 96 on page 291), the VR-Forces user cannot access chats specified for that force.

  A scenario-specific configuration file overrides the default channels. If you want force-specific channels, or one for all participants, you must define them in your configuration file. We did so in the example below.
- Channels must include one or more entities using entity names in the scenario for VR-Engage entities to be able to use the channel. (You cannot use the unit name to include all entities aggregated into that unit; you must use the individual entity names.) To include all entities, use the * wildcard. In the example below, the ALL channel includes all entities in the scenario, and the Blue Force and Red Force channels include all entities in the friendly and opposing forces, respectively.

This example is a configuration file for the VR-Engage_First_Experience scenario. It has a channel for only the paratroopers who start on the CH-47 Chinook CargRW 5, as well all the roles on that aircraft. (We also specified a force for that channel because, while only the specified entities can access that channel, we wanted VR-Forces users to be able to filter it by force.) It is a partial example; a real configuration file would have channels for the other groups in the scenario we planned for users to play in the simulation.

```
;; The filename is VR-Engage_First_Experience_ChatConfig.mtl
(chat-config
  (channels
    (channel
      (channel-name "CargoRW 5 Crew")
      (force 1)
      (entities
        (entity "Para 1")
```

```
          (entity "Para 2")
          (entity "Para 3")
          (entity "Para 4")
          (entity "CargRW 5")
        )
      )
      (channel
        (channel-name "ALL")
        (force 0)
        (entities
          (entity "*")
         )
      )
      (channel
        (channel-name "Blue Force")
        (force 1)
        (entities
          (entity "*")
         )
      )
      (channel
        (channel-name "Red Force")
        (force 2)
        (entities
          (entity "*")
         )
      )
    )
  )
```

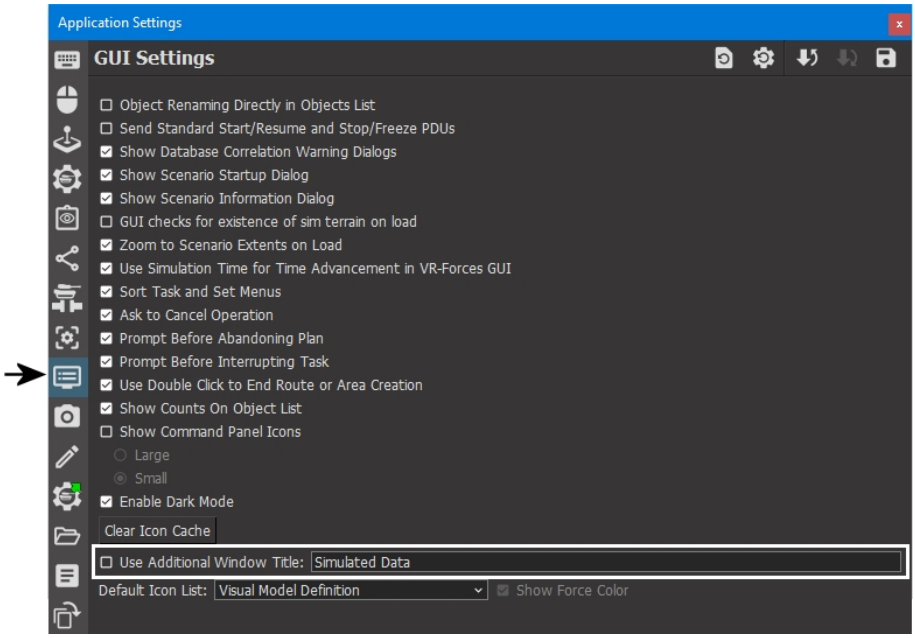
84.8. Adding Text to the Title Bar

VR-Forces displays the name of the scenario in the title bar. You can display additional text of your choice if you want to. The text is independent of the scenario. It is an application-wide setting.

➤ **To add text to the title bar:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 676).

Figure 676. GUI Settings page



3. Select the Use Additional Window Title check box.
4. Type the text you want to display in the text box.

84.9. Specifying Display Units

You can specify the units used to display coordinate system, distance, altitude, and other measurements. Display units are configured by measurement type, and can be specified for the application, simulation object platforms, and weather. This means that you can use different measurement units for different types of data. You can also specify the number of decimal places for measurement values. The available units of measurement for each data type are appropriate to that type.

When you change a measurement display unit, the changes take effect immediately. They are saved and applied the next time you run VR-Forces.

> The coordinate system information for the observer location in the Observer Information Panel and grid line coordinates, units, and precision are not affected by the settings described in this section.

The types of data and the display objects that they affect are:

- Coordinate system
- Distance
- Bearing and range
- Altitude
- Velocity (speed)

- Acceleration
- Rotational velocity
- Orientation (heading)

Table 75 describes the coordinate systems that you can specify.

Table 75. Coordinate systems supported

Coordinate system	Description
Latitude/Longitude (degrees:minutes:seconds)	Location displays as two position fields and one altitude field. Displays coordinates in the latitude and longitude using the geodetic WGS84 coordinate system. Each angle displays in degrees:minutes:seconds with seconds displaying base 10 fractional seconds.
Latitude/Longitude (degrees decimal minutes)	Location displays as two position fields and one altitude field. Displays coordinates in the latitude and longitude using the geodetic WGS84 coordinate system. Each angle displays in degrees:minutes with minutes displayed as a decimal.
Latitude/Longitude (decimal radians)	Location displays as two position fields and one altitude field. Displays coordinates in the latitude and longitude using the geodetic WGS84 coordinate system. Each angle displays in decimal radians.
Geocentric	Location displays as three position fields. Position coordinates are displayed in the geocentric coordinate system. The position fields are always in meters.
UTM	Location displays as two position fields and one altitude field. Position coordinates are displayed in the Universal Transverse Mercator system. The first position field displays the zone and the x location in meters. The second position field displays the y position in meters.
MGRS	Location displays as two position fields and one altitude field. Position coordinates are displayed in the Military Grid Reference System. The first position field displays the zone. The second position field displays the grid location. The precision controls the number of digits used in the grid display.
GARS	Location displays as one position field and one altitude field. Position coordinates are displayed in the Global Area Reference System. The position field shows the seven-character designation.
Database	Location displays as three position fields. Location displays using VR-Forces's current internal cartesian database system. The position fields are displayed using the current distance units.

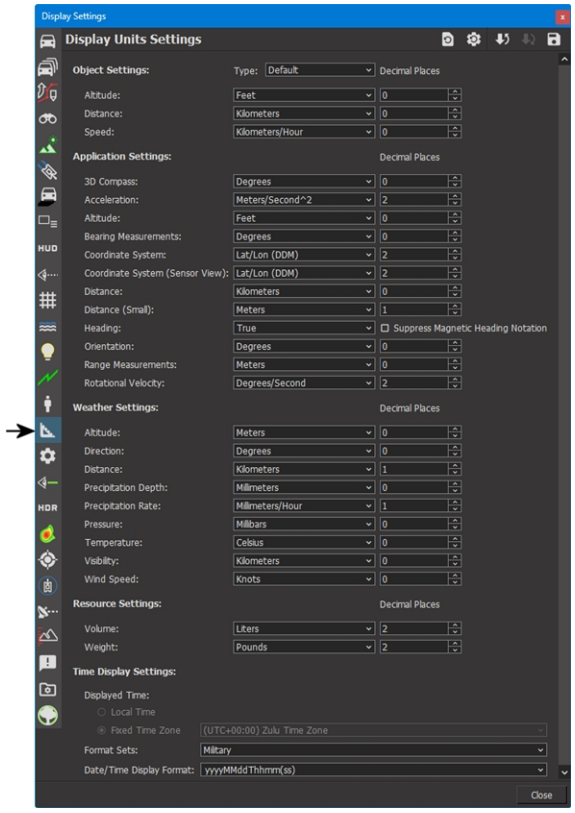
There are three options for Heading:

- True. True (geodetic) north is the direction pointing to the geographic location of the North Pole.
- Magnetic. Magnetic north is the point where the magnetic field points down.
- Grid. Grid north is the direction indicated by the vertical grid lines on ordnance survey maps.

➤ **To specify measurement units:**

1. Choose **Settings > Display**. The Display Settings dialog box opens.
2. Select the Display Units Settings page (Figure 677).

Figure 677. Display Units Settings page



3. To specify the measurement unit settings for objects, choose the type and then select the unit for each option from its list.
4. To specify the measurement unit settings for the application or weather, select the measurement unit you want for each option from its list.
5. Type a precision for each measurement unit in the corresponding box in the Decimal Places column, if applicable.
6. Specify the default time display format and zone.

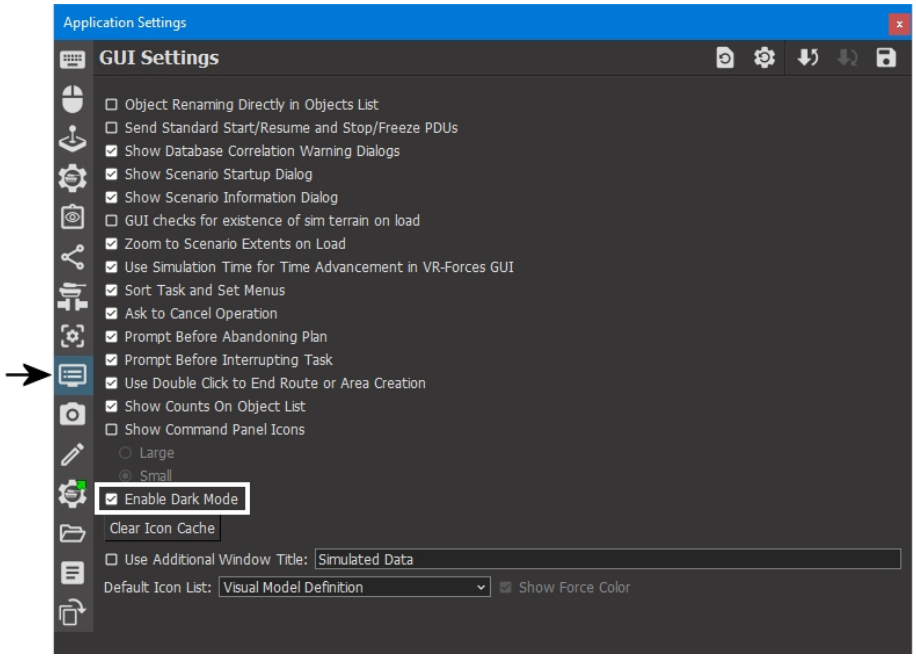
84.10. Disabling and Enabling Dark Mode

Dark mode is the default in VR-Forces's graphical user interface. Light mode, the former standard, is also available.

➤ **To switch the GUI from dark mode to light mode:**

1. Choose **Settings > Application**.
2. Select the GUI Settings page.

Figure 678. GUI Settings page



3. Clear the Enable Dark Mode check box. The window changes from dark mode to light mode. (Or select the Enable Dark Mode check box to see the window change from light to dark.)



85. Configuring Toolbars, Dialog Boxes, and Menus

This chapter explains how to specify which menus, menu options, toolbars, and dialog box pages display in the graphical user interface (GUI).

85.1. Introduction 1640
85.2. Configuring Menus and Menu Commands 1640
85.2.1 Alphabetizing the Task and Set Menus 1640
85.2.2 Creating New Menus 1641
85.3. Configuring Dialog Box Pages 1641
85.4. Configuring Context-Sensitive Menus 1643
85.4.1 Editing the Task Menu 1644
85.4.2 Editing the Set Menu 1645
85.5. Configuring Toolbars 1645
85.5.1 Creating Toolbars 1646
85.5.2 Specifying an Icon for a Toolbar Item 1646
85.6. Locking Main GUI Elements 1647

85.1. Introduction

VR-Forces displays menus, menu commands, toolbars, dialog box pages, and shortcut menus based on the information in an XML configuration file. The default file is *./appData/settings/vrfGui/default_guiConfiguration.gui*. You can edit this file or create alternative GUI configuration files. VR-Forces includes an example of a simple GUI configuration file, *./appData/settings/vrfGui/default_guiSimpleScenarioPlayer.gui*, that shows how you might create a highly modified GUI. You can specify an alternative GUI configuration file with the `--gui` command-line option when you start VR-Forces.

The elements in the configuration file are fairly self-explanatory. This chapter provides a brief description of the file.

For those GUI elements that can be rearranged, such as toolbars, the *.gui* file provides the initial organization of GUI elements. If a user rearranges toolbars, the new layout is saved in the settings file and overrides the *.gui* file.

85.2. Configuring Menus and Menu Commands

Menus are configured in the `<mainmenuconfiguration>` element. Each menu is configured in a `<mainmenu>` element. Menu commands are configured in `<menuitem>` elements. This example configures a menu bar with one menu that has two items, which are separated by a separator line:

```
<mainmenuconfiguration name="main_menu_configuration">
  <mainmenu menuname="DtFileMenu">
    <menuitem itemname="DtNewScenarioMenuItem"/>
    <separator separatorname="1"/>
    <menuitem itemname="DtExitItem"/>
  </mainmenu>
</mainmenuconfiguration>
```

The values for the `itemname` attributes must be valid IDs registered in VR-Forces code. *default_guiConfiguration.gui* has all of the valid IDs for menus and menu commands.

To remove a menu or menu command from the GUI, delete the applicable element from *default_guiConfiguration.gui* or create an alternative file that does not have the elements. To add menus or menu items, you must know the ID.

85.2.1 Alphabetizing the Task and Set Menus

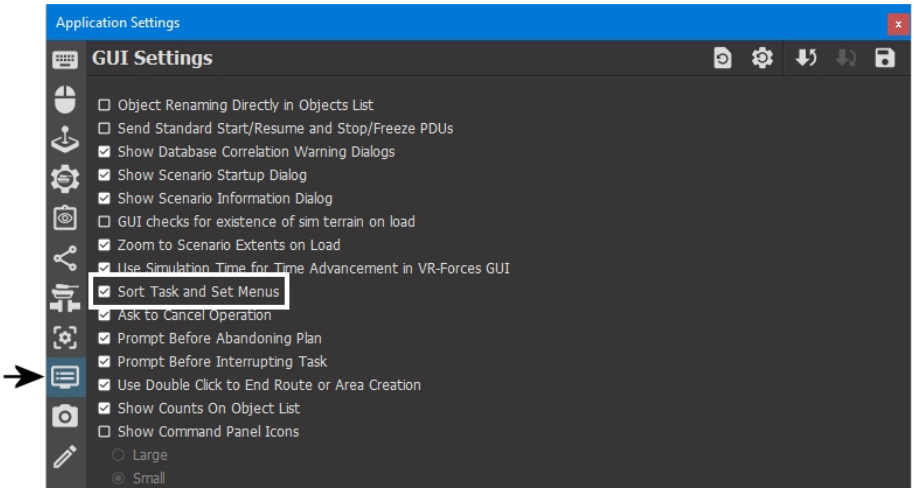
You cannot edit the Task and Set menus in *default_guiConfiguration.gui*. By default, VR-Forces alphabetizes the commands on the Task and Set menus and their submenus. This alphabetization overrides the menu ordering specified for scripted tasks and sets in their metadata. It also overrides menu placement specified for tasks and sets through the API.

If you want to, you can disable alphabetization and the commands will be listed according to script metadata and C++ code.

➤ **To enable or disable alphabetization of Task and Set menu commands:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the GUI Settings page (Figure 679).

Figure 679. GUI Settings page



3. Select or clear the Sort Task and Set Menus check box.

85.2.2 Creating New Menus

You can create new menus and submenus. To do so use the same syntax as the existing menu structure in *./appData/settings/vrfGui/default_guiConfiguration.gui*. For top level menus, add a **mainmenu** element. For submenus, use a **menu** element. For the menu name, specify the string you want to appear in the GUI. This example creates a menu called My File Menu at the top level. It has a submenu called Save Scenario Commands:

```
<mainmenuconfiguration name="main_menu_configuration">
  <mainmenu menuname="My File Menu">
    <menuitem itemname="DtNewScenarioMenuItem"/>
    <separator separatorname="1"/>
    <menu menuname="Save Scenario Commands">
      <menuitem itemname="DtSaveScenarioAsMenuItem"/>
      <menuitem itemname="DtSaveScenarioMenuItem"/>
    </menu>
    <menuitem itemname="DtExitItem"/>
  </mainmenu>
</mainmenuconfiguration>
```

85.3. Configuring Dialog Box Pages

Dialog boxes such as Display Settings and Terrain Settings have multiple pages for configuring different features. The GUI configuration file lets you disable or enable the pages on dialog boxes. The dialog boxes are configured as **<pagecollection>** elements inside the **<pagecollections>** element. This example configures the Application Settings dialog box:

```
<pagecollections name="page_collections">
    <pagecollection pagecollectionname="DtApplicationSettings"
title="Application Settings" location="8" visible="True" index="-1"
show="DtKeyMappingEditorPage" closable="True" floatable="True"
movable="True">
        <pageitem pageitemname="DtKeyMappingEditorPage"/>
        <pageitem pageitemname="DtMouseMappingEditorPage"/>
        <pageitem pageitemname="DtJoystickConfigurationPage"/>
         ...
        <pageitem pageitemname="DtScriptedTaskOptionsPage"/>
        <pageitem pageitemname="DtEntityBehaviorOptionsPage"/>
    </pagecollection>
</pagecollections>
```

In addition to settings dialog boxes, you can configure the pages on information dialog boxes, and the specific data on the pages. These are configured in **<informationpagecollection>** elements. The following example is part of the Information dialog box. The page is broken up into sections. The first section displays summary data. The second section has the individual pages and specifies the information on each page.

```
<informationpagecollections name="information_page_collections">
    <pagecollection pagecollectionname="DtEntityInformation"
title="Entity Information" location="9" visible="True" index="-1"
show="DtTaskStatePage" closable="True" floatable="True"
movable="True">
        <section sectionname="DtEntityCompactData"
maximumHeightIsMinimumHeight="true">
            <informationitem informationitemname="DtEntityIcon"
row="0" col="0" rowspan="2" width="50"/>
            <informationitem informationitemname="DtStateMarkingText"
row="0" col="1"/>
            <pageitem pageitemname="DtInformationIconBar" row="1"
col="1"     width="20">
                <informationitem
informationitemname="DtEmbarkedOnItem"/>
            </pageitem>
            <informationitem informationitemname="DtCreatePaletteName"
row="0" col="2"/>
            <pageitem pageitemname="DtSensorInformationPage" row="1"
col="2"/>
            <informationitem informationitemname="DtEntitySpeed"
row="0" col="3" width="128"/>
            <informationitem informationitemname="DtEntityAltitude"
row="1" col="3" width="128"/>
            <pageitem pageitemname="DtTaskStatePage" row="0" col="4"
width="128"/>
            <informationitem
informationitemname="DtObjectConsoleInformationStatusItem" row="1"
col="4"/>
        </section>
```

```
        <section sectionname="DtEntityExpandedData" collapsible="True"
tabbed="small" collapsed="True" title="Detailed Information">
            <pageitem pageitemname="DtTaskStatePage"/>
            <pageitem pageitemname="DtEntityInformationPage">
                <informationitem
informationitemname="DtEntityLocation"/>
                <informationitem informationitemname="DtEntitySpeed"/>
                <informationitem
informationitemname="DtEntityAltitude"/>
                <informationitem
informationitemname="DtEntityVelocity"/>
                <informationitem
informationitemname="DtEntityAcceleration"/>
                ...
                <informationitem
informationitemname="DtNavigationAreaPorts"/>
                <informationitem
informationitemname="DtEntityStateDRAlgorithm"/>
            </pageitem>
            <pageitem pageitemname="DtEntityAttributesPage"/>
            <pageitem pageitemname="DtResourceInformationPage"/>
            ...
        </section>
        <section sectionname="DtObjectConsoleSection" title="Object
Console" collapsible="true" collapsed="true"
maximumHeightIsMinimumHeight="true">
            <pageitem
pageitemname="DtObjectConsoleInformationWidget"/>
        </section>
    </pagecollection>
  ...
 </informationpagecollections>
```

Notice that by default, the DtTaskStatePage does not have any **<informationitem>** elements. The information on that page is generated dynamically. There are no static fields.

85.4. Configuring Context-Sensitive Menus

The Objects menu on the main menu bar and the context-sensitive, or popup, menus vary depending on which objects are selected. These menus are configured as a set of <rightclickmenu> elements. There are multiple versions of these elements to account for the different contexts in which a menu might be displayed, for example, one unit, multiple single entities, a mix of entities and control objects. This example is part of the context-sensitive menu for multiple entities:

```
 <rightclickmenu rightclickmenuname="Multiple_Entity">
   <popupmenu menuname="DtVrfRightClickContextMenu">
     <menu menuname="DtRightClickTaskMenu"/>
     <menu menuname="DtRightClickSetMenu"/>
     <menuitem itemname="DtTopLevelRightClickScriptedTasksSets"/>
```

```
      <menuitem itemname="DtVrfEntityPlanAction"/>
      <menu menuname="Manage Plan">
        <menuitem itemname="DtVrfEntityAbandonPlanAction"/>
        <menuitem itemname="DtVrfEntityRestartPlanAction"/>
        <menuitem itemname="DtVrfEntitySkipTaskAction"/>
      </menu>
        <menuitem itemname="DtVrfManageReactiveTasksAction"/>
        <menuitem itemname="DtVrfVrlinkObjectInformationItem"/>
        <separator separatorname="3"/>
        <menuitem itemname="DtVrfEditObjectAction"/>
        <menuitem itemname="DtVrfMoveObjectAction"/>
        <menuitem itemname="DtVrfCopyAction"/>
        <menuitem itemname="DtVrfDeleteAction"/>
        <separator separatorname="16"/>
      <menu menuname="Embarkation">
        <menuitem itemname="DtVrfEmbarkOnAction"/>
        <menuitem itemname="DtVrfDisembarkAction"/>
        <menuitem itemname="DtVrfDisembarkAllAction"/>
      </menu>
      <menu menuname="DtUnitDisplayEntityMenu">
        <menuitem itemname="DtVrfAggregateAsAction"/>
        <separator separatorname="AggSeparator1"/>
        <menuitem itemname="DtExpandAggregateMenuItem"/>
        <menuitem itemname="DtExpandAllAggregateMenuItem"/>
        <menuitem itemname="DtCollapseAggregateMenuItem"/>
        <menuitem itemname="DtCollapseAllAggregateMenuItem"/>
      </menu>
      <menu menuname="Decorations">
        <menuitem itemname="DtObjectSettingsAction"/>
      </menu>
    </popupmenu>
  </rightclickmenu>
```

Notice that the entries for the Task menu (DtRightClickTaskMenu) and the Set menu (DtRightClickSetMenu) do not have any `<menuitem>` elements. That is because these menus are themselves context-sensitive and therefore built dynamically.

85.4.1 Editing the Task Menu

The Task menu is a context sensitive menu built based on the platform type, controllers, scripts, and systems for a simulation object. Therefore, there is no all inclusive configuration file that you can edit to add and remove tasks. However, the file *./data/simulationModelSets/<sms>/gui/initialTaskMenuConfiguration.xml* is an SMS-specific set of tasks that are initialized when you load a scenario. If you remove a task from the list in this file, it will not be included on the Task menu. Removing a task in this way affects all simulation objects. If you edit this file, be sure to change the value for the **count** element to match your changes.

85.4.2 Editing the Set Menu

The Set menu is a context-sensitive menu built dynamically based on the controllers and scripts that define set data requests. While some set data requests are specific to particular platforms or object types, some are generically available to any platform that has a `set-data` controller in its platform file. This can result in some set data requests being available for objects for which they do not have much meaning. For example, you can set the ordered speed for a lamp post or a building.

You can cause VR-Forces to ignore set data requests when it builds the Set menu by adding an **ignore-sets** parameter to a `set-data` controller in an OPE file. This code shows the set-data controller in *AirBase.ope* in the AggregateLevel SMS:

```
(set-data
  (component-descriptor-type "set-controller-descriptor")
  (component-type  "cultural-feature-set-controller")
  (min-tick-period  -1.000000)
  (min-tick-period-variance  -1.000000)
  (tick-period-uses-real-time  False)
  (process-state-repository-name  "")
  (process-state-repository-type  "")
  (is-enabled  True)
  (debug-detail  False)
  (create-component  True)
  (ignore-sets "set-speed" "set-capability" "set-engagement-rules"
    "set-engagement-zones" "set-scan-sector" "set-current-speed"
    "set-altitude")
)
```

This method of editing the Set menu only applies to set data requests that are registered for in code, not scripted sets. The strings that you can add to **ignore-sets** are in *./include/vrftasks/radioMessageTypes.h*.

You can remove some set data requests from the menu for all simulation objects by editing *./data/simulationModelSets/<sms>/gui/initialSetMenuConfiguration.xml*. Be sure to change the value for the **count** element to match your changes.

85.5. Configuring Toolbars

You can configure use of the various toolbars provided with VR-Forces. For each displayed toolbar, you can configure which row it is in and which command icons are used. This example shows that the Scenario toolbar is in the top row:

```
<toolbarconfiguration name="toolbar_configuration">
  <toolbarrow toolbarname="Top_Row">
    <toolbar toolbarname="DtScenarioToolbar" visible="True"
         floating="False">
      <item itemname="DtNewScenarioMenuItem"/>
      <separator separatorname="DtScenarioToolbar_1"/>
      <item itemname="DtLoadScenarioMenuItem"/>
      <item itemname="DtLoadBatchScenarioMenuItem"/>
      <separator separatorname="DtScenarioToolbar_2"/>
```

```
        <item itemname="DtSaveScenarioMenuItem"/>
        <separator separatorname="DtScenarioToolbar_3"/>
        <item itemname="DtCloseScenarioMenuItem"/>
      </toolbar>
    </toolbarrow>
  </toolbarconfiguration>
```

85.5.1 Creating Toolbars

Since toolbars are just buttons that reference existing menu options, you can create your own toolbars as easily as you add, delete, or edit the toolbars provided with VR-Forces. Simply add a `toolbar` element in one of the `toolbarrow` elements.

If you want to add tasks or sets to the Task or Set toolbars, or create a new toolbar, specify the `itemname` using the task or set ID as listed in *./include/vrfGuiConstants/vrfMenuIds.h*. If you want to add a scripted task to a toolbar, the `itemname` is its script ID plus "ToolbarItem". For example, if you add the Place IED scripted task to the Task toolbar, it would look like this:

```
<toolbar toolbarname="DtTaskToolbar" visible="True" floating="False">
  <item itemname="DtVrfTaskMoveToActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveToLocationActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveAlongRouteActionToolbarItem"/>
  <item itemname="DtVrfTaskMoveToAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskPatrolObjectsActionToolbarItem"/>
  <item itemname="DtVrfTaskPatrolRouteActionToolbarItem"/>
  <item itemname="DtVrfTaskEmbarkActionToolbarItem"/>
  <item itemname="DtVrfTaskDisembarkActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyHeadingActionToolbarItem"/>
  <item itemname="DtVrfTaskFlyHeadingAndAltitudeActionToolbarItem"/>
  <item itemname="DtVrfTaskFireAtTargetActionToolbarItem"/>
  <item itemname="Place_IEDToolbarItem"/>
</toolbar>
```

85.5.2 Specifying an Icon for a Toolbar Item

If you add an item to a toolbar, you can associate an icon with the toolbar button.

To associate an icon with a scripted task, specify it in the Scripts dialog box. For details, see "36.2.2 Specifying a Menu Icon" on page 753.

For tasks or sets, create an SVG file for the light mode GUI and put it in *./data/Icons*, and also a file for dark mode and put it in *./data/Icons/DarkMode*. The file for both modes must have the same name.

The file should use the menu ID of the task or set (from *./include/vrfGuiConstants/vrfMenuIds.h*) with the DtVrf text and Action text stripped from it. For example, the icon for DtVrfTaskMoveToAltitudeAction is *TaskMoveToAltitude.svg*. The icon for DtVrfSetHeadingAction is *SetHeading.svg*.

> The DtVrfSetLaserCodeAction action is an exception to the file naming convention. You need to name the icons SetLaser.svg.

85.6. Locking Main GUI Elements

VR-Forces plug-ins can modify menus, dialog boxes, and other GUI features. You can lock GUI settings for first level elements (elements under the root element). This feature is used in *./appData/settings/vrfGui/default_guiSimpleScenarioPlayer.gui*. The syntax is:

```
<mainmenuconfiguration name="main_menu_configuration" locked="True">
  ...
</mainmenuconfiguration>
```

Given this configuration, none of the menus on the main menu could be modified.



86. Creating and Editing Key Mappings

This chapter explains how to edit key mappings and create new key mappings.

86.1. Introduction to Key Maps 1650
86.2. The Key Mapping Editor 1650
86.2.1 Binary Key Mappings 1651
86.3. Editing a Key Map 1651
86.3.1 Adding a Key Mapping 1651
86.3.2 Changing a Key Mapping 1653
86.3.3 Deleting a Key Mapping 1653
86.4. Using Combined Key Mappings 1654
86.5. Filtering the Function List 1655
86.6. Creating a Key Map 1655
86.7. Deleting a Key Map 1655

86.1. Introduction to Key Maps

When you move the observer, you are actually executing functions that control the observer's movement. For example, the Move Observer Forward function moves the observer forward in observer coordinates. The Move Level Observer Forward function moves the observer forward in level observer coordinates.

The various movement functions are mapped to the keys on the keyboard and to mouse buttons and the mouse wheel.

VR-Forces comes with key mappings that we think will meet the needs of most VR-Forces users. However, you can edit the default keyboard mappings and create new mappings of your own with the Key Mapping Editor.

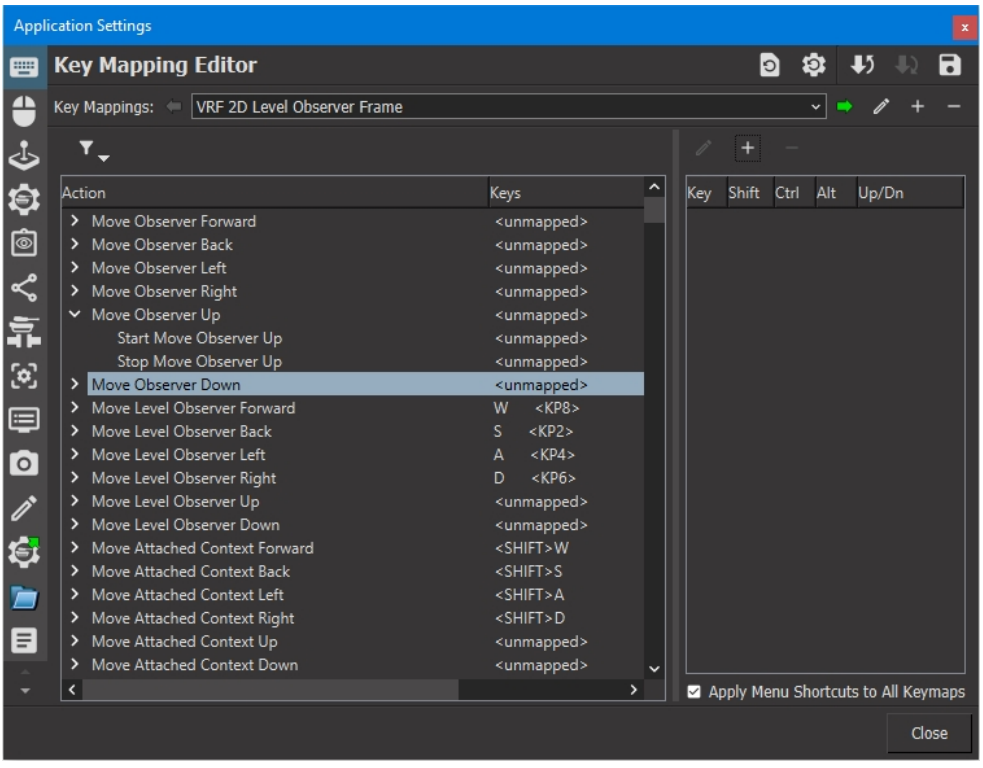
86.2. The Key Mapping Editor

The Key Mapping Editor (Figure 680) lists all of the functions that you can control from the keyboard and the keys to which they are mapped. Some functions are mapped to more than one key or key combination. A key can be mapped to more than one function (combined mappings). You can:

- Edit a function's key mappings.
- Add new mappings.
- Delete mappings.

Changes are saved automatically.

Figure 680. Key Mapping Editor



86.2.1 Binary Key Mappings

Many navigation functions require you to press and hold a key. For example, you press and hold **W** to move the observer forward. (By contrast, some keyboard actions are toggles or unary functions, such as pressing period to attach to the next simulation object.) For these actions, VR-Forces needs to keep track of when the key is pressed down (start the action) and when it is released (stop the action). To a human user, a function like Move Observer Forward is experienced as one action. But the computer experiences it as two actions. Therefore, key mappings must take this into account. The Key Mapping Editor shows the key mappings for these functions as paired functions that are subordinate (children) to the primary function (the parent), as shown in the Move Observer Up entry in Figure 680.

86.3. Editing a Key Map

The Key Mapping Editor lists all of the VR-Forces functions that you can control from the keyboard.

➤ **To edit a key map:**

1. Choose **Settings > Application**. The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 680 on the previous page).
3. Select the function whose key mappings you want to edit.
4. Edit the key mappings. as described in:
   - "86.3.1 Adding a Key Mapping" below.
   - "86.3.2 Changing a Key Mapping" on page 1653.
   - "86.3.3 Deleting a Key Mapping" on page 1653.
5. Optionally, export the key map to a file.
6. Click OK.

86.3.1 Adding a Key Mapping

If a function is not mapped to a key, it is marked **<unmapped>**. You can add a mapping to unmapped functions and you can add additional key mappings to functions that are already mapped. If you add a key mapping to a binary function, the Key Mapping Editor automatically applies it to the child (UP and DN) functions. You can add key mappings to the child functions of binary mappings. When you add a key mapping to a child function, it is not applied to its paired function or the parent function.

ⓘ In some cases, an unmapped function is unmapped because it is not meaningful in the observer frame for the key mapping. For example, the level observer movement functions are not meaningful in the observer frame key mappings.

➤ **To add a key mapping:**

1. Optionally, filter the function list. For details, see "86.5. Filtering the Function List" on page 1655.
2. Select the function to which you want to add a key mapping.
3. Click the Add button (+) above the Key Map window (Figure 681). The Key Entry dialog box opens (Figure 682).

Figure 681. Key Mapping Editor

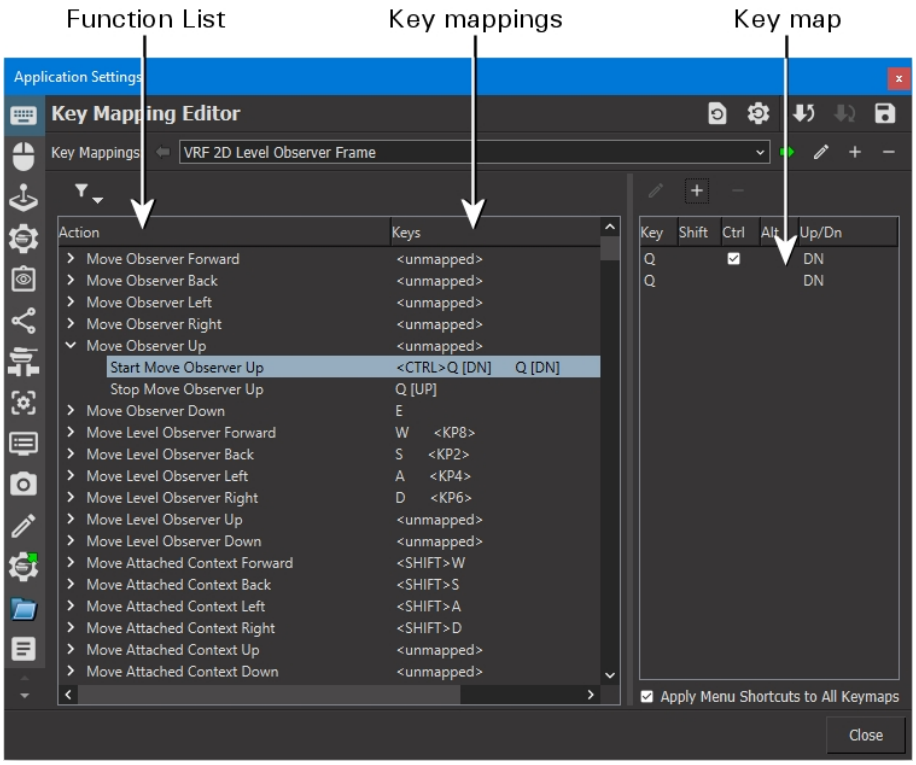
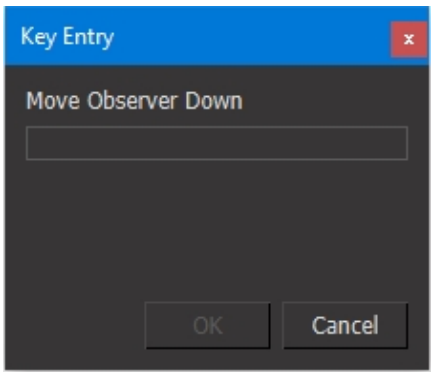


Figure 682. Key Entry dialog box



4. On the keyboard, press the key or key combination (such as **Ctrl**+key or **Shift**+key) that you want to map to this function.

5. Click OK. (If you choose a key combination that is already in use, VR-Forces notifies you and asks how to proceed. For details about duplicate key mappings, see "86.4. Using Combined Key Mappings" on the next page.) The mapping is added to the function and is listed in the Keys column of the function list and in the Key Map window.

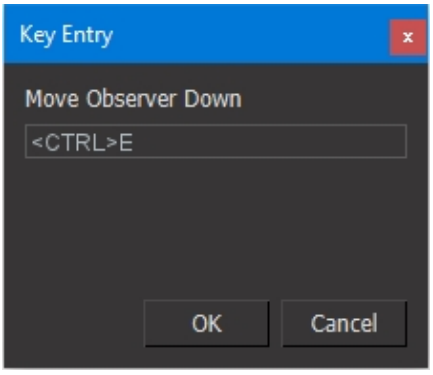
86.3.2 Changing a Key Mapping

If a key mapping is assigned to a binary function, the key is automatically assigned to the child functions. You cannot edit the key mappings for the child functions. You can only change the parent's key mapping. However, if you add a key mapping to a child function, you can edit it.

➤ **To change a key mapping:**

1. Select the function whose key mapping you want to change. Its key mappings are listed in the Key Map window (Figure 681).
2. In the Key Map window, select the key mapping that you want to change.
3. Click the Edit Key Mapping button (). The Key Entry dialog box opens (Figure 683). It shows the current key mapping.

Figure 683. Key Entry dialog box



4. On the keyboard, press the key or key combination (such as **Ctrl**+*key* or **Shift**+*key*) that you want to map to this function.
5. Click OK. The mapping changes.

86.3.3 Deleting a Key Mapping

➤ **To delete a key mapping:**

1. Select the function whose key mapping you want to delete. Its key mappings are listed in the Key Map window (Figure 681).
2. In the Key Map window, select the key mapping that you want to delete.
3. Click the Delete button () above the Key Map window. The mapping is deleted. The function list is updated.

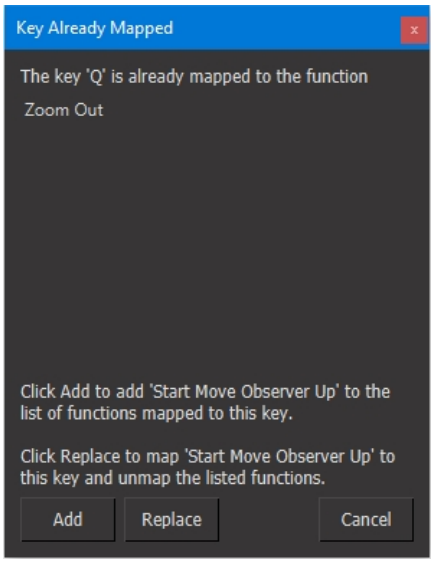
86.4. Using Combined Key Mappings

You can assign a key combination to more than one function. This allows you to combine the behaviors of two or more functions into one key combination. For example, the **Z** key detaches the observer from the attached simulation object. The **Spacebar** resets the observer view to the default view. Suppose that whenever you detach from a simulation object, you want the observer to return to the default view. You could map the Reset function to **Z**. **Z** is now mapped to two functions. When you press **Z**, VR-Forces detaches from the attached simulation object and moves the observer to the default view.

➤ **To manage combined key mappings:**

1. Add the appropriate key mapping to a function.
2. Click OK. The Key Already Mapped dialog box opens (Figure 684). It lists the functions that already use this key mapping.

Figure 684. Key Already Mapped dialog box



3. If the functions listed are the ones with which you want to share this key mapping, click Add. The key mapping is added to the function.
4. If you do not want to share this key mapping with other functions and want to use it for the function that you are editing, click Replace. The key mapping is applied to the function you are editing and removed from the functions listed in the dialog box.
5. If you do not want to remove the key mapping from existing functions and do not want to share it with the other functions, click Cancel. Presumably you wanted a unique key mapping for the function you are editing and did not realize you chose one that was already in use. Choose a different key mapping for this function.

86.5. Filtering the Function List

You can filter the function list in the Key Mapping Editor. Functions are grouped thematically.

➤ To filter the function list, click the Filter button ( ) and select an option on the list.

86.6. Creating a Key Map

➤ **To create a key map:**

1. Choose **Settings > Application**.The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 681 on page 1652).
3. Click the Add button ( ). The Create New Key Map dialog box opens.
4. Type a name for the new key map.
5. Click OK.
6. Add key mappings for the movement actions as described in "86.3.1 Adding a Key Mapping" on page 1651.

86.7. Deleting a Key Map

➤ **To delete a key map:**

1. Choose **Settings > Application**.The Application Settings dialog box opens.
2. Select the Key Mapping Editor page (Figure 681 on page 1652).
3. In the key mapping list, select the key mapping that you want to delete.
4. Click the Delete button ( ).
5. Confirm that you want to delete the selected key map.



XV. Appendices

The appendices contain supplementary information.



A. Troubleshooting

This appendix describes solutions to common issues. If your problem is not covered here, check the MAK ONE Product Support Knowledge Base: https://www.mak.com/knowledge.

If you require assistance, visit MAK's Support Portal and provide any information about your issue. See "Reporting an Issue Using the .callstack.log and .dmp Files" on page 99.

A.1. Using the Simulation Engine Console 1660
A.2. Performance and Display Issues 1660
A.2.1 Features You Can Disable to Improve Performance 1660
A.3. The Fault Tolerant Heap 1661
A.4. Linux CentOS 8 Memory Issue 1661
A.5. Seamless Scenes with Multiple Channels 1661
A.6. Feature Indexing for Model Layers 1662

A.1. Using the Simulation Engine Console

You can use the simulation engine console (also known as the back-end console) for debugging, running diagnostics, and otherwise testing VR-Forces. To learn about the available commands, enter `help` in the console.

A.2. Performance and Display Issues

There are a few issues that require you to add the application name to the *BlackApps.dat*:

- Nahimic can severely impact performance and frame rates because it interacts with certain hardware capabilities, vantage leverages, in an extremely detrimental way.
- Under certain circumstances, opening VR-Forces on a second display prevents the program from starting.
- In addition, launching from the main display and then moving the mouse cursor to the other monitor causes the program to freeze and not respond.
- Sometimes, selecting an SMS to open in the Simulation Object Editor causes the program to hang.
- Nahimic can also sometimes cause VR-Forces to crash.

If you experience any of these issues, edit the Nahimic *BlackApps.dat* file and add the application name (for example, vrfLauncher.exe or simulationObjectEditor.exe) to the list. You can search *C:\ProgramData* for *BlackApps.dat* to locate it. This file might be found in *C:\ProgramData\A-Volute\A-Volute.Nahimic\Modules\Scheduled\Configurator*, or *C:\ProgramData\A-Volute\A-Volute.28054DF1F58B4\Modules\ScheduledModules\Configurator*, or something similar.

A.2.1 Features You Can Disable to Improve Performance

There are a number of features that can affect frame rate. To improve performance, disable the ones that you do not need.

- Clouds. Reduce the clouds in the scenario. To reduce the clouds, select a predefined environment condition in the Weather page of Environment Conditions. See "11.4. Selecting an Environment Condition" on page 333.

  You can also adjust the clouds using the Advanced Weather settings, as described in "11.2.1 Setting the Sky Cover, Wind Direction and Speed" on page 324.
- Lighting. If you do not need to light the scene with a high level of realism, disable high dynamic range (HDR) in the High Dynamic Range Settings page of Display Settings. See "52.9. Using High Dynamic Range Lighting" on page 1097.
- Shadows. If you do not require high quality shadows, disable horizon-based ambient occlusion (HBAO) in the Observer Settings page of Display Settings. See "6.8.2 Using the Ambient Occlusion Effect" on page 209.

  You can also disable shadows entirely, or display them for only certain simulation objects. See "52.11. Displaying Shadows (3D Only)" on page 1100.

- Ocean. For scenarios that do not include the ocean, you can disable the dynamic ocean in the Observer Settings page of Display Settings. See "11.8.1 Enabling Marine Effects" on page 342.
- Trees. If you do not need SpeedTrees for your scenario, disable them in the SpeedTree settings page of Display Settings (Figure 417 on page 1102).
- Anti-aliasing. Reduce anti-aliasing by using the `--antiAliasing` *`level`* command-line option, setting *`level`* to either `2` or `0`.
- Terrain layers. If there are layers in the terrain that you do not need for your scenario, disable them in the Terrain Editor Panel. See "Enabling and Disabling Layers" in the *Adding Content to MAK ONE Applications Reference Manual*.

A.3. The Fault Tolerant Heap

The Fault Tolerant Heap (FTH) is a subsystem on Windows that can affect software developers and testers. If a software application crashes many times—for example, in the process of programming and debugging that application—Windows adds it to the FTH. This is significantly slower than the normal heap and can have a negative impact on the performance of the application as a result.

For more information and to learn possible solutions, see https://docs.microsoft.com/en-us/windows/win32/win7appqual/fault-tolerant-heap.

A.4. Linux CentOS 8 Memory Issue

There is a known problem with VR-Forces and CentOS 8 running out of memory. When the OS runs out of memory or starts using the swap memory, it can become unresponsive. Here are some potential solutions:

- Upgrade the Linux kernel to at least version 5.7. We have found that the default kernel of 4.18 becomes unresponsive when the swap is being used but the 5.7 kernel does not have this problem.
- Adjust the terrain being loaded in VR-Forces to have fewer osgEarth layers and simpler models to reduce the memory consumption.
- Add more RAM to your computer.

A.5. Seamless Scenes with Multiple Channels

When you use multiple channels to display a scene because you want more than a 45 degree FOV, as described in "Chapter 83. Display Engine and Window Management" on page 1597, you should disable HDR (High Dynamic Range) and Ambient Occlusion (HBAO+). If you do not, there are visual issues where the channels overlap. For example, if there is a light point that is positioned on the line between the two channels, the halo effect is shown unevenly across the two channels.

➤ **To disable HDR and HBAO+:**

1. In VR-Forces, choose **Settings > Display**.
2. Select the Observer Settings page (Figure 45 on page 210).

3. Select the observer that is assigned to the channels for displaying greater FOV, for example, Observer 1.
4. Make sure the observer mode is the 3D mode that you are using, for example, Stealth or Out-the-Window.
5. Clear the Ambient Occlusion (HBAO+) check box.
6. Select the HDR page.
7. Clear the Enable High Dynamic Range check box.

A.6. Feature Indexing for Model Layers

Feature indexing for model layers in osgEarth terrains is not supported by VR-Forces and must be disabled on all model layers to avoid performance issues and the possibility of crashes:

```
<model>
  <feature_indexing enabled="false"/> <!-- include this
  with every model section for VR-Vantage. If enabled,
  feature_indexing causes performance issues and crashes
  on Linux -->
<!-- rest of model layer setup here-->
</model>
```



B. MTL Syntax

This chapter describes the syntax of VR-Forces MTL files.

B.1. Introduction 1664
B.2. MAK Technologies Lisp (MTL) Files 1664
B.2.1 Using Environment Variables in an MTL File 1665
B.2.2 Specifying Lists of Lists 1665
B.2.3 Using Conditional Statements 1665

B.1. Introduction

VR-Forces uses many different configuration files to manage element definitions, model mapping, sim engine configurations, menus, and so on. Almost all of the configuration files for the GUI are XML files that are created and updated through the Display Settings dialog box, Terrain Settings dialog box, and the other GUI configuration dialog boxes. Although you could edit these files with text editors, we recommend that you not edit them by hand and we do not document their contents and syntax. When you save settings, they are saved to files with the correct extension.

> Developers may need to create and edit configuration files using the VR-Forces Graphical User Interface (Front-End) API. Please see the *VR-Forces Developers Guide* and the class documentation for details.

The sim engine configuration files mostly use the MAK Technologies LISP (MTL) syntax. The major sim engine configuration file is *vrfSim.mtl*. It is described here, in "Appendix B. MTL Syntax" on the previous page. Other files are described throughout this manual in the appropriate section.

B.2. MAK Technologies Lisp (MTL) Files

Many of the configuration files for MAK products are ASCII text files that use MAK Technologies LISP (MTL) to encode configuration information. MTL files have the extension *.mtl*. You can edit them in any text editor. Be sure that your text editor uses the proper end-of-line characters for your platform.

Most parameter entries take the format:

```
(setq parameter_name value)
```

For example:

```
(setq EnableWireFrame 1)
```

Or:

```
(setqb parameter_name value)
```

The setqb syntax indicates a bound variable.

To comment out a line, precede the text with two semi-colons (`;;`). You can also add a comment at the end of a line by preceding it with two semi-colons.

In most cases, the parameter names are known to the MAK application and the MTL commands simply assign values to them. However, you can use the `setq` command to create arbitrary symbolic names or aliases that are then used in later commands. For example, in the following two commands, the symbolic name VaporTrail is assigned to an OpenFlight file. Then the symbolic name is used in the `trail-map` command. The name VaporTrail has no intrinsic meaning. It is given meaning by the `setq` command, which allows it to be used later in the `trail-map` command.

```
  (setq VaporTrail (list "./../data/models/makTrails/VaporTrail"
"VaporTrail.flt" 6.0 1.5 1.5 4.0 1.5 0.5))
  (trail-map (list 1 2 -1 0 -1 -1 -1)(list VaporTrail 0.0 0.0 0.0))
```

In addition to the `setq` command, a commonly used command is the `load` command. This command instructs the application to load the file specified, for example:

```
(load mtl-path "params.mtl")
```

B.2.1 Using Environment Variables in an MTL File

If you want to use an environment variable in an MTL file, use the syntax:

```
(setqb parameter_name (getenv (quote env_var)))
```

For example:

```
(setqb disDestAddr (getenv (quote DEST_ADDR)))
```

B.2.2 Specifying Lists of Lists

The example of setting a symbolic name in a previous section uses a list. You can also specify a list of lists, for example:

```
(trail-map (list 1 2 -1 -1 -1 -1 -1)
  (list vaporTrail -8.0 -3.0 0.0)
  (list vaporTrail -8.0 3.0 0.0)
)
```

A parameter entry can also have multiple individual lists:

```
(trail-map (list 1 2 -1 0 -1 -1 -1)(list VaporTrail 0.0 0.0 0.0))
```

B.2.3 Using Conditional Statements

MTL commands can be conditional. Conditional statements are used frequently when parameters are different for DIS and HLA. A simple conditional statement has the format:

```
(if (condition) (statement_to_evaluate))
```

If you have more than one statement to evaluate, use a sequence block:

```
;; HLA specific parameters
(if (equal HLA 1)
  (sequence
    (setqb fomMapperLib "")
    (setqb fomMapperInitData "")
    (setqb rprFomVersion 1.0)
    (setqb pathToSublistFile "sublist.mtl")
    (setqb ignoreAdvisories 0)
    (setqb fedLookahead 1.0)
  )
)
```



C. The vrfSim.mtl Configuration File

This chapter describes the *vrfSim.mtl* file.

C.1. The vrfSim.mtl Configuration File Parameters 1668

C.1. The *vrfSim.mtl* Configuration File Parameters

The file *./appData/settings/vrfSim/vrfSim.mtl* configures sim engines. Table 76 describes the parameters in *vrfSim.mtl*. The Description column includes the command-line argument equivalent for parameters that have them.

Table 76. *vrfSim.mtl* parameters

Parameter	Description
additionalSystemScriptsDirectory	Directories other than the default script directory where VR-Forces should look for system scripts.
appNumber	Specifies the application number. Default: 3001. Command line: `(--appNumber | -a)` *number*
assertOnBlockingTerrainCalls	When set to true (1), causes blocking terrain calls to assert in debug mode if the simulation is running. Blocking terrain calls can cause long ticks in a running simulation which can have very negative effects on the models. This parameter can help developers find places in the code that are making the offending calls. Note that calling the blocking form of any terrain intersection function causes an assertion; it is not necessary for the call to actually block and even non-paging implementations will assert. This is to make it easier to find potential problems regardless of the particular terrain or location being used. Default: 0. For more information, see API documentation.
automaticallyCreateGlobalEnvironment	The default behavior is false (0), ensuring that the application does not create a global environment object when a user creates a scenario and disables the global environment. When set to true (1), the application creates a global environment object if one does not exist. This is not generally recommended and can cause problems.
batchScenarioFileName	Specifies a batch file to run. For use with only one sim engine. Command line: `(-B | --batchScenarioFileName)` *batch_file*

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
blockOnAsynchronousOperations	Applies only to scenarios that are set to run in a fixed frame clock mode (Fixed-Frame Best-Effort or Fixed-Frame Run-To-Complete). If enabled, scenarios that are fixed frame rate will block on asynchronous operations (for example, terrain, feature, path planning, and navigation operations). Default: 0.
cgfDispatcherReceivePort	Specifies the port to be used by the *DtCgfDispatcher's* file transporter to receive scenario objects and plan files from the *DtVrfRemoteController*.
clockMode	Configuration in Windows for absolute timestamping. When set to 0, uses the performance counter, which is a highly accurate internal timer (default behavior). When set to 1, synchronizes with the system clock on every tick. Command line: `--clockMode` *mode* (Windows only.) Use Absolute Time Stamps must be enabled.
countryCodesMappingFile	The file VR-Forces uses to import country codes to map to object type enumerations.
daemon	Runs vrfSim as a daemon process (Linux only.) vrfSim does not try to read characters from a console, and forks a background process on startup. It exits cleanly on receipt of the SIGTERM, SIGQUIT, SIGINT signals. Default: Off. Command line: `--daemon`
datumShiftX **datumShiftY** **datumShiftZ**	These parameters, along with the **spheroidSemiMajor** and **spheroidSemiMinor** parameters, let you change the reference ellipsoid used for geocentric to geodetic conversions. For more information, see Choosing a Reference Ellipsoid in *VR-Link Developers Guide*. These parameters are used only if **VRF_useCustomMapDatum** is set to 1.
defaultParameterDatabasePath	Specifies the location to search for the object parameter database if the full path is not specified in a scenario file.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
defaultTerrainDatabasePath	Specifies the location to search for the terrain database if the full path is not specified in a scenario file.
diGuyAnimationsFile	Specifies the DI-Guy animations file to load.
disableParallelPublisherTick	Controls whether to tick publishers in parallel (0) or synchronously (1). Set to tick synchronously if you experience issues with ticking in parallel, for example, when using the HLA protocol and the RTI implementation is not thread safe. Default: 1
disableParallelTick	If enabled (1), VR-Forces ticks thread safe components in parallel. If disabled (0), all components are ticked serially in the main thread. Default: 0
doNotUseConsole	Disables (1) or enables (0) writing of output to the console. Default: 0.
enableLogFileTimestamps	If enabled, puts timestamps in front of each line in for all log files.
enableNavigationLabDebugging	Enables debugging using the Navigation Lab application. A connection is made on an available port, starting with 4888, to enable Navigation Lab to connect to the simulation engine.
entVrfDataTimeThreshold	This has been deprecated.
forceParameterDbReload	If set to 1 (default), forces the sim engine to reload the object parameter database (OPD) each time a scenario is loaded. If set to 0, if a scenario is started that uses the same SMS as the previous one (for example, in a rewind), the SMS is not reloaded.
gamewareMemorySize	Set the working memory size limit used internally by GameWare when calculating paths. Size is in MB. Increasing this limit can allow for path plans on larger navigation areas by more entities simultaneously. Note: Multiple working memory blocks are allocated with this limit, so this is not an overall limit for Gameware.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
gamewareQueryTimeBudget	Set the time budget for GameWare navigation queries per frame. This is the amount of time GameWare is given to perform tasks such as path planning each simulation frame. The higher the value, the more responsiveness path planning will be, but the slower the VR-Forces simulation frame rate will be. Default: 1.0 ms.
hostAddressString	Specifies the host address. This is usually the same as the device address.
loadAllNavigationDataOnTerrainLoad	Loads all of the navigation data when the scenario loads, rather than waiting until an entity needs to use that data (either when it is placed in a navigation area or moves into one). If set to 1, navigation loads the data at the same time you load or create a scenario. Please be aware this may negatively impact performance. If it is set to 0, the navigation data loads only when an entity exists in the navigation area. Default: 0.
loadPluginIfVersionMismatch	Specifies whether (1) or not (0) to load a plug-in if there is a mismatch between how the plug-in was built and how VR-Forces was built. Mismatches can be build type (release versus debug), operating system, or protocol (DIS, HLA Evolved.) Allowing a plug-in to load if there is a mismatch can cause VR-Forces to be unstable.
logFrameRateStatistics	Enables (1) or disables (0) logging of frame rate statistics to a file. Default: 0
maxAsynchronousTerrainThreads	Sets the maximum number of threads that will spawn when aysnchronous terrain processing jobs are required. Setting it to 0 allows unlimited threads. Default: 7.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
maxDrainInputTime	Sets the maximum time to wait in drainInput() before returning. For the MAK RTI, the default value drains the entire input stream before returning. For the Pitch RTI, the default value reads a single input from the input stream, so set to a value other than -1 to process more information. Use with DIS or HLA.
maxNavigationRegenerationThreads	Sets the maximum number of threads that can be running at a given time regenerating GameWare navigation data. If you have many available CPUs and are using dynamic terrain, increasing this number speeds up the regeneration of navigation data due to changes in the terrain. If you are more limited in CPU availability, increasing this value may negatively affect VR-Forces performance. Default: 2.
msdlSIDCMappingFile	The file VR-Forces uses to map between SIDC codes and object type enumerations when it imports MSDL files.
navigationRegenerationDelay	Sets the delay (in seconds) between the occurrence of an event that triggers navigation data regeneration and the start of the regeneration. Decreasing this improves the responsiveness of navigation to dynamic terrain changes; however, it increases the likelihood that navigation data will generate multiple times when multiple events result in navigation changes in a particular area. Default: 2.5
navigationRegenerationFileDirectory	Specifies a location where the application will save files of the navigation data generated at run time. You can use the files for debugging, such as when navigation data is not generated as expected. No files are generated when this parameter is not enabled.
notifyLevel	Sets the notification level (0–4) for information and warning messages. Default: 2. Command line: `(--notifyLevel | -n)` *notification_level*

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
numCallbackThreads	Specifies the number of threads the sim engine should create when simulating entities in the scenario.
numNetworkCallbackThreads	When running the network in a thread, specifies the number of threads the sim engine should create when simulating entities in the scenario.
numberOfNavQueueProcessingThreads	Sets the number of threads that are spawned to process navigation queue queries. Adding more threads decreases the length of time it takes for entities to perform path planning, however it uses up additional CPU resources. Setting the value to 0 will run all navigation queue query processing in the main thread. Default: 2
objectConsoleNotifyLevel	Specifies the notification level (0–4) for messages sent to the object console. Default: 1.
pluginsDirectory	Specifies the directory in which configuration (MTL) files for plug-ins are located. The directory must end with a slash, for example, *./plugins/*
publishZeroVelocityWhenPaused	When set to 1, the default, VR-Forces sets simulation object velocities to 0 when it is paused so that simulation objects do not continue to dead reckon. To publish the current velocity, set to 0.
regenerateNavigationDataAtRuntime	Enables navigation data to be regenerated at runtime based on changes in dynamic terrain. This may have a negative effect on performance as it requires spawning threads to perform the regeneration. However, it allows entities to correctly navigate around and through the changes in the terrain. Default: 1.
rejectMismatchedScenarioMessages	If true (the default) reject mismatched scenario run messages based on unique scenario run ID.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
reportedHostAddressString	Specifies how VR-Forces reports its IP address to another VR-Forces application for the purpose of file transfers. It is only important when you have a GUI or remote controller on one side of a router and a simulation engine on the other, and there is some kind of network address translation occurring whereby they have to connect to one another using an IP address other than the one they recognize themselves at locally.
reuseEntityIdentifiersOnScenarioLoad	If true, reuse object identifiers that might be part of the scenario. If the object identifier saved in the scenario objects file does not currently match the site/app of the sim engine loading that object, then the user will be warned and the object will be assigned a new entity identifier that is compliant with that site/app. This will not apply to simulation object groups or scenario import. They are always assigned new entity identifiers.
scenarioFileName	Specifies the scenario file to load. Command line: `(-L | --scenarioFileName) "`*`scenario_pathname`*`"`
sendBackendLogToNetwork	Enables (1) or disables (0) sending simulation engine log messages to the GUI.
sendMessagesInNetworkProcess	Specifies whether to send messages in a network thread (1) or immediately (0). Default: 1
sessionId	Specifies the session ID. Command line: `(-i | --sessionId)` *`ID`*
setDeadReckoningAlgorithmToStaticOnPause	When the simulation is paused, set the dead reckoning algorithm for aggregates and entities to static. If enabled and **publishZeroVelocityWhenPaused** is disabled, the objects in a VR-Forces run simulation show current values for velocity, speed, acceleration, and so on.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
setMainThreadToHighPriority	If set to 1, sets the main thread to THREAD_PRIORITY_TIME_CRITICAL in Windows. Using this option on a single core processor will probably cause deadlock. (Windows only.)
simMgmtPduProcessLevel	Specifies how simulation management PDUs are processed: 1 – Processes PDUs sent to a specific sim engine. Wildcards are ignored. 2 – Processes only wildcarded PDUs. Ignores those sent to specific sim engines. 3 – Processes all PDUs. 4 – Processes any PDUs that match the sim engine's address. Wildcards are allowed.
simulationModelSet	Specifies the default simulation model set or sets. If specifying multiple SMSs, separate them by a semicolon.
siteId	Specifies the site ID. 0 = use internal default. Command line: `(--siteId | -s)` *`ID`*
spheroidSemiMajor **spheroidSemiMinor**	These parameters, along with the datumShift parameters, let you change the reference ellipsoid used for geocentric to geodetic conversions. For more information, see Choosing a Reference Ellipsoid in *VR-Link Developers Guide*. These parameters are used only if **VRF_useCustomMapDatum** is set to 1.
startInRunMode	Specifies that the simulation start in run mode (1), rather than pause mode (0). Default: 0.
startMinimized	If true (1), the sim engine minimizes at startup. Default: 0
statusUpdateHeartbeat	Specifies how often, in seconds, the sim engine status heartbeat is sent.
suppressDataAvailableException	Prevent terrain interface non-blocking intersect calls from throwing an exception if the dataAvailable flag is not checked.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
targetFrameRate	Specifies the frame rate for the current frame above which the sim engine will sleep, yielding CPU time to other processes. This option applies only to variable frame mode. For more information, see "6.11. Tuning the Target Frame Rate" on page 215. Default: 30.
taskVisualizationPublicationMode	Specifies whether to publish task visualizations and how to publish them: 0 = Never publish task visualizations. 1 = Standard mode. The VR-Forces GUI will request task visualizations only for objects for which they are enabled. 2 = Always publish task visualizations, regardless of whether they are being displayed in the GUI. Can be useful for logging purposes.
terrainDatabase	Specifies the terrain database to load. This parameter is overridden by the scenario file if one is specified in **scenarioFileName**. Command line: `(-T | --terrainDatabase)` *`terrain_database`*
tickPublishersDuringEntityTick	Controls whether to tick publishers in the network thread (1) or during the advance frame (0). Ticking the publishers in the network thread (while entities are ticking) works when the application ticks publishers synchronously and it will not impact the ticking of entities. Default: 1
useAbsoluteTimeStamps	If set to 1, use absolute timestamps. If set to 0, use relative time stamps. Default: 0.
useCustomMapDatum	Allows modification of the reference ellipsoid used for geocentric to geodetic conversion. If you enable this parameter, there are several sub-parameters to configure. For details, see the comments in the file.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
useDrawControlForTaskVisualizations	If true, publish task visualizations as VR-Vantage draw control objects instead of their standard publication method. This allows task visualizations to be displayed in any VR-Vantage based application; however the VR-Forces GUI will no longer be able to filter task visualizations by entity.
HLA Parameters	**For details, see "5.5. Command-Line Options for HLA Federations" on page 190.**
destroyFedExec	Specifies whether (1) or not (0) the *DtExerciseConn* destructor should try to destroy the federation execution. Default: 1.
execName	The federation execution name. Command line: `(--execName | -x)` *`exec_name`*
expectedLocalEntityCount	Specifies the number of entities that a single sim engine will simulate. When using the HLA 1516 Evolved protocol, the application uses this to determine the number of HLA object names that it will preregister.
federateName	The federate name. Command line: `(--federateName | -N)` *`federate_name`*
federateType	Lets you specify the federate type for HLA Evolved.
fedFileName	Specifies the FED file name. Command line: `(--fedFileName | -F)` *`fedfile_name`*
fomMapperLib	A FOM Mapper library. Command line: `(--fomMapperLib | -f)` *`libname`*
fomMapperInitData	An initialization string. Command line: `--fomMapperInitData` *`data`*
localSettingsDesignator	Specifies the local settings designator for HLA Evolved.
mimModule	Specifies a MIM (MOM and Initialization) Module for HLA Evolved.
minDrainInputTime	The minimum time to stay in `drainInput()` before returning.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
rprFomVersion	The RPR FOM version. (0.5, 0.7, 0.8, 1.0, 2.00*nn*) Command line: `--rprFomVersion` *`version_number`*
runInTimeManagementMode	Enables (1) or disables (0) use of time management. Default: 0
sendFedTime	Enables (1) or disables (0) sending and receipt of interactions in timestamp order. Default: 0.
useAdvisories	Enables (1) or disables (0) HLA advisory messages. Default: 1
useSingleNetworkThreadWhenRti ThreadSafetyNotGuaranteed	When using the MAK RTI and HLA 1516 Evolved, you can specify that VR-Forces alter the configuration to ensure that access to RTI ambassador is thread safe, limiting the sim engine to a single thread (1). If you are certain access to your RTI ambassador is thread safe, set this parameter to 0. Default: 1
DIS Parameters	**See "5.6. Command-Line Options for DIS Exercises" on page 192.**
additionalDestinationAddresses	Specifies additional destination addresses to send to. Accepts multiple IP addresses separated by semicolons (`;`).
addMulticastAddr	The multicast address. Command line: `-S`
appIdRange	Specifies the application ID range to use to identify simulation objects generated by VR-Forces.
defaultHeartbeatThreshold	Specifies the DIS heartbeat threshold, in seconds. A heartbeat is sent when the threshold is reached, even if no data has changed.
defaultObjectTimeout	Specifies the DIS object timeout threshold, in seconds. If VR-Forces does not receive a heartbeat for an object for the time period specified, it removes the object from the exercise.
destAddrString	Destination address for point-to-point communications mode. Command line: `(--destAddrString | -A)` *`address`*
deviceAddress	Specifies the address of the network card to use for UDP traffic.

Table 76. vrfSim.mtl parameters (continued)

Parameter	Description
disPort	The port number. Command line: `(--disPort | -P)` *port*
exerciseId	The exercise ID. Command line: `(--exerciseId | -x)` *ID*
heartbeatVariance	When a simulation object is created, a random value between -**heartbeatVariance** and +**heartbeatVariance** is added to the heartbeat. This spreads heartbeats over time instead of them all being at the same time.
maxDrainInputReads	The maximum number of reads to make in drain input before returning
mcastTtl	Specifies the multicast time to live. Command line: `--mcastTtl` *time*
recvBufferSize	Specifies the receive buffer size, as a positive value. A value of -1 means use the system default. Command line: `--recvBufferSize` *size*
sendBufferSize	Specifies the send buffer size, as a positive value. A value of -1 means use the system default. Command line: `--sendBufferSize` *size*
subnetMask	When the destination address is not specified ("0.0.0.0"), the subnet mask is used in conjunction with the network interface address to determine the destination broadcast address. If neither is specified (subnet mask is blank), the network interface's default broadcast address is used.
suppressSelfReflect	Prevent VR-Forces from reflecting DIS PDUs.
useAsyncIO	Enables (1) or disables (0) use of asynchronous I/O, which can help reduce dropped packets under heavy network load. Default: 0. Command line: `--useAsyncIO`
useIpv6	Use the IPV6 protocol.



D. Object Parameters

This appendix describes the high-level parameters in the Simulation Model Set. It also has reference material for local and remote subcomponents and object type enumerations.

D.1. The Structure of a Simulation Object Parameter 1682
D.2. The Parameter Type Hierarchy 1682
D.2.1 Vrf Base Object Parameters 1682
D.2.2 Vrf Object Parameters 1687
D.2.3 Moving Object Parameters 1687
D.2.4 Parameter Types for Entity Platforms 1688
D.3. Object Type Enumerations 1688
D.3.1 Object Super-type 1688
D.3.2 AggregateKind 1689
D.3.3 ObjectKind 1689
D.3.4 Unit Category 1689
D.3.5 Unit Specific 1690
D.3.6 Point Object Category 1691
D.3.7 Linear Object Category 1691
D.3.8 Areal Object Category 1691
D.3.9 Interaction 1691

D.1. The Structure of a Simulation Object Parameter

Many of the files in the object parameter database uses MAK Technologies LISP (MTL). (For details about MTL, see "B.2. MAK Technologies Lisp (MTL) Files" on page 1664.) A parameter block has the form:

```
(block_name
  ( parameter_name value)
  ...
  (parameter_name
    (sub-parameter value)
  )
  and so on
)
```

Entries have these requirements:

- The block_name can be anything that you like. It must not start with a number.
- The parameter_name must be exactly as documented (except that in some cases, a parameter may contain an arbitrary length list of sub-parameters, for which the parameter names are not used – these will be noted). Parameter names must not contain spaces, and must not start with a digit (or a dash followed by a digit, for example, -1).
- The first parameter in any parameter block must be **parameter-type**.
- You can include comments in the file. A comment begins with a semicolon(;) and continues to the end of the line.
- All numerical parameters use standard MKS values: meters, kilograms, seconds, radians, liters, and so on.
- Real constants must begin with a digit and contain a decimal point. For example, 0.5 and 0. are acceptable, but .5 and 0 are not.



.entity files use XML format, not MTL.

D.2. The Parameter Type Hierarchy

Figure 626 on page 1472 illustrates the hierarchy of parameter types. In this section, we describe the parameters associated with each **parameter-type**.

D.2.1 Vrf Base Object Parameters

All object parameters inherit from the parameter type **vrf-base-object-param**. The parameters associated with vrf-base-object-param are potentially applicable to any type of object (ground, rotary-wing, control object, and so on). In addition to parameters, each object has an object type and parameter type, as described in Table 77.

Table 77. Object and parameter type

Parameter	Description
object-type	(Required) 8-digit object type. Not actually a parameter. Specifies the key used to look up the entry. For more information, see "75.1.2 Object Types" on page 1469.
parameter-type	(Required) One of (string): • "vrf-base-object-param" • "vrf-object-param" • "moving-object-param" • "fixed-wing-entity-param" • "ground-vehicle-param" • "rotary-wing-entity-param" • "missile-param" • "surface-entity-param" • "subsurface-entity-param" • "human-param" • "aggregate-object-param" • "cultural-feature-param" • "bomb-param" • "global-environment-param" • "circle-object-param" • "point-object-param" • "scripted-movement-object-param" • "tactical-smoke-param" The value must be in quotation marks.

Local Object Subcomponent Parameters

The **local-objects** parameter specifies the types of subcomponents to create for locally simulated objects. You can omit optional items, or set them with an empty string “”. Table 78 lists the subcomponent parameters and their possible values.

Table 78. Local object subcomponent parameters

Subcomponent parameter	Description
state-repository	(Required) Maintains the state information for the object. An object that has components (sensors, controllers, and actuators) for simulation sets and retrieves information in the repository. Some simulation components expect a specific kind of state repository. Specify the state repository type that most closely matches the kind of simulation object you expect to simulate. Consider these types: • "vrf-object-base-state-repository": Used for the base object in the object parameter database. • "vrf-object-state-repository": Derived from vrf-object-base-state-repository, it adds the concepts of resources, sector of responsibility, engagement rules, and targeting. Because there are more simulation object-specific state repository types that are derived from this type, this kind of repository is not typically specified. • "vrf-overlay-object-state-repository": For tactical graphics. • "vrf-moving-object-state-repository": For any object that you expect to simulate movement. Units in entity-level scenarios have this kind of state repository. The more simulation object-specific repository types that follow inherit this kind of state repository, so choose one of those when simulating a specific type of individual entity. • "ground-entity-state-repository": For ground vehicles, such as tanks, trucks, dismounted infantry. • "rotary-wing-entity-state-repository": For rotary-wing aircraft. • "fixed-wing-entity-state-repository": For fixed-wing aircraft. • "missile-entity-state-repository": For missile and rocket entities. • "surface-entity-state-repository": For surface entities. • "subsurface-entity-state-repository": For subsurface entities. • "human-state-repository": For lifeform entities. • "vrf-aggregate-state-repository": For units. • "cultural-feature-state-repository": For cultural features that are created as entities.

Table 78. Local object subcomponent parameters (continued)

Subcomponent parameter	Description
network-interface	(Optional) The network interface handles the publication of the object over the network. It acts as an interface between a simulation object's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of simulation object you are simulating. Consider these types: • "aggregate-local-entity-net-interface": For units. • "pseudo-aggregate-local-entity-net-interface": For units that have subordinate simulation objects that are separately simulated. • "individual-local-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. • "local-environmental-net-interface": For simulation objects that are best represented as an environmental process. Tactical graphics use this type of network interface.
net-interface-min-tick-period	(Optional) Specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's network interface.
net-interface-min-tick-period-variance	(Optional) Specifies an amount within which the **net-interface-min-tick-period** can vary. Defined as: plus or minus rand(**net-interface-min-tick-period**/2)
component-manager	(Optional) A component manager contains the lists of sensor, controller, and actuator components used to simulate the behavior of a given object (typically simulation objects). Expected value: "component-manager".
plan-manager	(Optional) The simulation object plan manager. If you want a simulation object to be able to execute a plan, it must have a plan manager.
task-manager	(Optional) If a component manager is specified for an object, then you must specify a task manager. The task manager works in conjunction with the component manager to manage the processing and message handling associated with execution of a task. Expected value for individual entities or units: "local-task-manager".

Remote Object Subcomponent Parameters

The **remote-objects** parameter specifies the types of sub-components to create for remote objects. Remote objects are those objects simulated by other applications on the network (including other VR-Forces applications). You can omit optional items, or set them with an empty string "". Table 79 lists the subcomponent parameters and their possible values.

Table 79. Remote object subcomponent parameters

Subcomponent parameter	Description
state-repository	(Required) Maintains the state information for the object. Unlike the local-object entry, it is not as important to specify the closest match based on type. For entities, the "vrf-moving-object-state-repository" is sufficient. A more specific state repository may be specified. However, some of the simulation object-specific data may not be set. May be one of these types: • "vrf-object-state-repository": Derived from vrf-object-base-state-repository, it adds the concepts of resources, sector of responsibility, engagement rules, and targeting. Because there are more simulation object-specific state repository types that are derived from this type, this kind of repository is not typically specified. • "vrf-overlay-object-state-repository": For tactical graphics. • "vrf-moving-object-state-repository": For any object that you expect to simulate movement, including individuals and units. • "ground-entity-state-repository": For ground vehicles, such as tanks, trucks, dismounted infantry. • "rotary-wing-entity-state-repository": For rotary-wing aircraft. • "fixed-wing-entity-state-repository": For fixed-wing aircraft. • "missile-entity-state-repository": For missile and rocket entities. • "human-state-repository": For lifeform entities. • "vrf-aggregate-state-repository": For units. • "cultural-feature-state-repository": For cultural features that are created as entities. • "surface-entity-state-repository": For surface entities. • "subsurface-entity-state-repository": For subsurface entities.

Table 79. Remote object subcomponent parameters (continued)

Subcomponent parameter	Description
network-interface	(Optional) The network interface is the interface between the remote network (VR-Link) representation of the object and its state repository. It acts as an interface between a simulation object's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of simulation object you are simulating. May be one of these types: • "vrf-aggregate-remote-entity-net-interface": For remote units. • "vrf-individual-remote-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. • "vrf-remote-environmental-net-interface": For simulation objects that are best represented as an environmental process. Tactical graphics use this type of network interface.
net-interface-min-tick-period	(Optional) Specifies the shortest period of time, in seconds, that can elapse between ticks of a simulation object's network interface.
net-interface-min-tick-period-variance	(Optional) Specifies an amount within which the **net-interface-min-tick-period** can vary. Defined as: plus or minus rand(**net-interface-min-tick-period**/2)
plan-manager	Do not specify a plan manager for remote objects. (Give it the value "" (empty string). If you specify a plan manager, it is ignored.)
component-manager	Do not specify a component manager for remote objects. (Give it the value "" (empty string). If you specify a component manager, it is ignored.)
task-manager	(Optional) Do not specify a task manager for remote objects. (Give it the value "" (empty string). If you specify a task manager, it is ignored.)

D.2.2 Vrf Object Parameters

The parameter type **vrf-object-param** inherits from **vrf-base-object-param** and is the parent of the moving object parameter type. Vrf object parameters include resources, engagement rules, and target acquisition time. The **vrf-object-param** parameter type represents an intermediate class in the parameter hierarchy. It is not usually used to represent objects in VR-Forces. You will probably want to instantiate a derived type, (for simulation objects) or you will want to specify vrf base object parameters (for case of control objects.)

Control objects have the parameter type **vrf-base-object-param**.

D.2.3 Moving Object Parameters

Moving objects have the parameter type **moving-object-param**. Moving objects inherit parameters from vrf objects and vrf base objects. The **moving-object-param** parameter type is used for objects that are capable of movement, such as simulation objects.

Parameters include support points, maximum speed, maximum reverse speed, turning radius, maximum slope, ordered speed, maximum acceleration, maximum deceleration, and fuel efficiency. Units are moving objects and have this parameter type. All of the platform-specific parameter types derive from this one.

D.2.4 Parameter Types for Entity Platforms

Table 80 lists the parameter types used by each entity type.

Table 80. Parameter types for simulation objects

Platform	Parameter Type	Inherits from
Ground vehicle	**ground-vehicle-param**	**moving-object-param**
Fixed-wing	**fixed-wing-entity-param**	**moving-object-param**
Human	**human-param**	**moving-object-param**
Rotary-wing	**rotary-wing-entity-param**	**moving-object-param**
Surface	**surface-entity-param**	**moving-object-param**
Subsurface	**subsurface-entity-param**	**moving-object-param**
Cultural feature	**cultural-feature-param**	**moving-object-param**
Missile	**missile-param**	**moving-object-param**
Unit	**aggregate-param**	**moving-object-param**
Unit Container	**moving-object-param**	**vrf-object-param**

D.3. Object Type Enumerations

This section lists the object type enumerations as listed in the header file *objType-Enums.h*. It is presented here primarily for customers who have purchased VR-Forces, but not the toolkit. Therefore we have departed from the typical header file format for readability.

D.3.1 Object Super-type

The header file *objTypeEnums.h* contains the VR-Forces extensions to the DIS/RPR FOM enumerations. The enumerations specify individual, unit, and other object types. The sim object super-type is an enumeration used in the *DtObjectType* class (an extension of the *DtEntityType* class). It extends the DIS/RPR FOM entity type enumeration by specifying the sort of object associated with the *DtEntityType*. Because object types are not guaranteed to be unique across all of the possible individual and unit types, the extra enumerated field provides a way to uniquely identify all objects in a VR-Forces simulation.

VR-Forces uses the value 3 in the object super-type for all of the units that it simulates.

Other	0
Individual	1
Unit	2

Disaggregated Unit	3
Aggregated Unit	4
Interaction	5

D.3.2 AggregateKind

The enumeration for *DtAggregateKind* should start with 0, but VR-Forces starts with 10, to ensure that simulation object types will be unique.

Other	10
MilitaryHierarchy	11
CommonType	12
CommonMission	13
SimilarCapabilities	14
CommonLocation	15
DtSimArtPartKind	20

D.3.3 ObjectKind

Extension of entity kind to represent non-simulation object objects, such as control objects (control points, routes, phase lines, obstacles, and so on.)

DtPointObjectKind	16
DtLinearObjectKind	17
DtArealObjectKind	18
DtCircleObjectKind	19
DtTextObjectKind	20

D.3.4 Unit Category

VR-Forces extends the enumeration to include *DtAggregateCategoryForce*, which represents the force level. In the list, Force, Team, Squad, and Section are extensions to the standard.

Other	0
IndividualVehicle	1
Element	2
Platoon	3
Battery	4
Company	5

Battalion	6
Regiment	7
Brigade	8
Division	9
Corps	10
Force	11
Team	12
Squad	13
Section	14

Unit Subcategory

The Subcategory field enumerations for Kind field = Military Hierarchy specify the Echelon Type of the unit. Not all of the echelon types are applicable to each of the categories.

In this list, ForceFriendly, ForceOpposing, ForceNeutral, and ForceOther are extensions to the DIS/RPR FOM enumerations:

Other	0	ArmyAviationFixedWing	13
CavalryTroop	1	ArmyAviationRotaryWing	14
Armor	2	ArmyAttackHelicopter	15
Infantry	3	AirCavalry	16
MechanizedInfantry	4	ArmorHeavyTaskForce	17
Cavalry	5	MotorizedRifle	18
ArmoredCavalry	6	MechanizedHeavyTaskForce	19
Artillery	7	CommandPost	20
SelfPropelledArtillery	8	CEWI	21
CloseAirSupport	9	TankOnly	22
Engineer	10	ForceFriendly	23
AirDefenseArtillery	11	ForceOpposing	24
AntiTank	12		

D.3.5 Unit Specific

The Specific field enumerations for Kind field = Military Hierarchy specify whether the unit contains a headquarters, and is enumerated as follows.

Noheadquarters	0
ContainsHeadquarters	1

D.3.6 Point Object Category

Subcategories for use with the Kind field = *DtPointObjectKind*.

Other	0
ControlPoint	1
TargetPoint	2
Marker	3

D.3.7 Linear Object Category

Subcategories for use with the Kind field = *DtLinearObjectKind*.

Other	0
PhaseLine	1
Route	2

D.3.8 Areal Object Category

Subcategories for use with the Kind field = *DtArealObjectKind*.

Other	0
ControlArea	1
Obstacle	2

D.3.9 Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



E. Systems and System Usage

The tables in this appendix provide information about the systems provided with VR-Forces and list the simulation objects that use them.

E.1. VR-Forces Systems 1694
E.2. Entity-level systems and configured scripted tasks 1694
E.3. Aggregate-level systems and configured scripted tasks 1702

E.1. VR-Forces Systems

To see a list of systems for each SMS with a description, a list of simulation objects that use it, and the types of ammunition used (if applicable), load the SMS in the Simulation Object Editor and view the General tab for each system.

Some scripted tasks are only available to simulation objects that have a particular system. See "E.2. Entity-level systems and configured scripted tasks" below and "E.3. Aggregate-level systems and configured scripted tasks" on page 1702.

E.2. Entity-level systems and configured scripted tasks

Table 81. Systems and configured scripted tasks, EntityLevel.sms

System	Scripted Tasks
Aggregate Mechanized Maneuver	move-to-waypoint-path-plan, move-to-location-path-plan, test-squad-reconnoiter
Aggregate Vehicle Crew Maneuver	set-assigned-transport, set-lock-embarkation, auto_move_to_ location, crew-operate-cargo-door, crew-pick-up-at-location, crew-drop-off-at-location, crew-provide-mobile-overwatch, crew-provide-stationary-overwatch, move-to-waypoint-path-plan, move-to-location-path-plan, set-dismounted, set-mounted-crew, mount-vehicle-crew, dismount-vehicle-crew
Air Cushion	maneuver-in-formation, set_freeze_fuel_levels, set_freeze_ movement, set_navigation_priority, ground-vehicle-move-to, move_to_position_and_heading, turn_to_heading_along_arc
Air Refueling Receiver Boom	refuel_from, deploy_refueling_probe, stow_refueling_probe
Air Refueling Receiver Drogue	refuel_from, deploy_refueling_probe, stow_refueling_probe
Cargo Door	Open_Cargo_Door, Close_Cargo_Door
Sliding Door	Open_Sliding_Door, Close_Sliding_Door
Towed Array Decoy	extend_towed_array_decoy, retract_towed_array_decoy
Air Drop Capable	airdrop_from_aircraft, parachute_to_target
Air Drop	Air_Drop, ,, airdrop_along_route, base_airdrop
Animal Four Legged	maneuver-in-formation, move-to-location-path-plan, move-to-waypoint-path-plan, set_navigation_priority, follow_along_offset_ route, set_freeze_movement, auto_move_to_location, set-unloaded, ground-vehicle-move-to, move_to_position_and_ heading, turn_to_heading_along_arc

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Animal Herd Movement	wander-herd, set_navigation_priority, set_freeze_movement
Animal Subsurface	set_freeze_movement, subsurface-move-to
Animal Two Legged Default	set_navigation_priority, follow_along_offset_route, set_freeze_ movement, human-move-to
Anti-Submarine Missile (Vertically Launched)	Launch_ASW_Missile
Background Pedestrian Generator	create_background_pedestrians
Boom Air Refueling Supplier	set_cargo, establish_refueling_station, Deploy_Refueling_Boom, Stow_Refueling_Boom, stow_port_refueling_drogue, stow_ starboard_refueling_drogue, deploy_port_refueling_drogue, deploy_starboard_refueling_drogue
Boom Wingtip Drogue Air Refueling Supplier	set_cargo, establish_refueling_station, Deploy_Refueling_Boom, Stow_Refueling_Boom, stow_port_refueling_drogue, stow_ starboard_refueling_drogue, deploy_port_refueling_drogue, deploy_starboard_refueling_drogue
Car Door	open_car_door, close_car_door, set_open_car_door, set_close_car_ door
Cargo Air Drop Capable	cargo_parachute_to_target
Centerline Boom Drogue Attachment Air Refueling Supplier	set_cargo, establish_refueling_station, Deploy_Refueling_Boom, Stow_Refueling_Boom, stow_port_refueling_drogue, stow_ starboard_refueling_drogue, deploy_port_refueling_drogue, deploy_starboard_refueling_drogue
Cruise Missile	set_freeze_fuel_levels, set_freeze_movement
DDG Embedded Support Craft (LAMPS/RHIB)	Embedded_Sector_SAR
Dumb Bomb Dynamics	set_freeze_fuel_levels, set_freeze_movement
Fast Roping	Deploy_Ropes, FastRopeInsertion, Rope_Embarkation, Rope_ Disembarkation, rope_sling_load_pickup, rope_sling_load_dropoff
AMRAAM Missile Launcher	attack_aircraft
Meteor Missile Launcher	attack_aircraft
MICA RF/EM Missile Launcher	attack_aircraft

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Sidewinder Missile Launcher	attack_aircraft
Fixed Wing Disaggregated Movement	set_freeze_fuel_levels, fixed-wing-unit-move-to
Fighter Jet	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object, fly_loiter_pattern_fwa, fly_circular_arc, fly_to_position_on_circle, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, crank_left, crank_right, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, land_at_airport_near_location, land_on_runway, takeoff_on_runway, land_at_aerodrome, land_on_route, takeoff_on_route, fixed-wing-move-to
Heavy Plane	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object, fly_loiter_pattern_fwa, fly_circular_arc, fly_to_position_on_circle, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, crank_left, crank_right, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, land_at_airport_near_location, land_on_runway, takeoff_on_runway, land_at_aerodrome, land_on_route, takeoff_on_route, fixed-wing-move-to
High Maneuverability Fighter	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object, fly_loiter_pattern_fwa, fly_circular_arc, fly_to_position_on_circle, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, crank_left, crank_right, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, land_at_airport_near_location, land_on_runway, takeoff_on_runway, land_at_aerodrome, land_on_route, takeoff_on_route, fixed-wing-move-to
Fixed Wing VTOL	Fixed_Wing_Takeoff, Fixed_Wing_Land, orbit_object, fly_loiter_pattern_fwa, fly_circular_arc, fly_to_position_on_circle, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, crank_left, crank_right, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, land_at_airport_near_location, land_on_runway, takeoff_on_runway, land_at_aerodrome, land_on_route, takeoff_on_route, fixed_wing_vertical_takeoff, fixed_wing_vertical_land, fixed_wing_hover, fixed-wing-move-to
Folding Wing	Fold_Wings, Unfold_Wings
Ground Casualty Evacuation	ground-casualty-evacuation

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Ground Convoy Movement	set_freeze_fuel_levels, set_freeze_movement, set_prefer_roads, ground-vehicle-unit-move-to, ground-unit-move-to, push-ordered-speed, pop-ordered-speed
Ground Disaggregated Movement	set_freeze_fuel_levels, set_freeze_movement, transition_into_formation, take_formation, set_prefer_roads, ground-vehicle-unit-move-to, ground-unit-move-to, push-ordered-speed, pop-ordered-speed, maneuver-along
Ground Disaggregated Movement For Higher Echelon Units	set_freeze_fuel_levels, set_freeze_movement, set_prefer_roads, ground-unit-move-to, push-ordered-speed, pop-ordered-speed
Tracks	maneuver-in-formation, move-to-location-path-plan, move-to-waypoint-path-plan, set_navigation_priority, follow_along_offset_route, set_freeze_fuel_levels, set_freeze_movement, auto_move_to_location, set-unloaded, ground-vehicle-move-to, move_to_position_and_heading, turn_to_heading_along_arc, push-ordered-speed, pop-ordered-speed, move_next_to_target
Tracks/Amphibious	maneuver-in-formation, move-to-location-path-plan, move-to-waypoint-path-plan, set_navigation_priority, follow_along_offset_route, set_freeze_fuel_levels, set_freeze_movement, set-unloaded, ground-vehicle-move-to, move_to_position_and_heading, turn_to_heading_along_arc, push-ordered-speed, pop-ordered-speed
Wheels (off road)	maneuver-in-formation, move-to-location-path-plan, move-to-waypoint-path-plan, set_navigation_priority, follow_along_offset_route, set_freeze_fuel_levels, set_freeze_movement, auto_move_to_location, set-unloaded, ground-vehicle-move-to, move_to_position_and_heading, turn_to_heading_along_arc, push-ordered-speed, pop-ordered-speed, move_next_to_target
Wheels (road)	maneuver-in-formation, move-to-location-path-plan, move-to-waypoint-path-plan, set_navigation_priority, follow_along_offset_route, set-unloaded, ground-vehicle-move-to, move_to_position_and_heading, turn_to_heading_along_arc, push-ordered-speed, pop-ordered-speed, move_next_to_target
Throw Grenades	ThrowGrenade, ThrowGrenadeTarget
Handheld Javelin Launcher	launch_javelin_at_target, launch_atgm_at_target
Handheld UAV	orbit_object, set_freeze_fuel_levels, set_freeze_movement
Homing Torpedo Capability (Aerial and Surface Forward Launched)	Delayed_homing, Delayed_homing_ASW, Launch_Torpedo_Salvo

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Homing Torpedo Capability (Subsurface Forward Launched)	Delayed_homing, Delayed_homing_ASW, Launch_Torpedo_Salvo
Human Disaggregated Movement	circle_around_location, find_cover_unit, find_firing_positions, move_between_cover_to_location_unit, set_navigation_priority, occupy_firing_positions, unit_travel_offset_routes, set_freeze_ movement, human-unit-move-to, ground-unit-move-to, navigation_enabled, unit-ground-follow
Flee From Explosions	bhave-Flee_From_Explosion
Flee From Threats	flee_from_threat
Human Default	embark_using_rope_ladder, embark_using_fast_rope, disembark_ using_rope_ladder, disembark_using_fast_rope, set_navigation_ priority, follow_along_offset_route, set_freeze_movement, human-move-to-location-then-posture, crouch-along-route, dismount-vehicle-transport, human-move-to-location-heading-posture, navigation_enabled, move_next_to_target
IFF Code Changer	iff-code-changer
IFV Script Enable	ifv-suppress-enemies
Javelin Missile Dynamics	set_freeze_fuel_levels, set_freeze_movement
KN-02 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
KN-08 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Laser Guided Missile Dynamics	set_freeze_fuel_levels, set_freeze_movement
Lifeform Non Lethal Damage	non_lethal_damage_effect
Lifeform Suppression	Suppressed_Crouch, Suppressed_Prone
M203 Grenade Launcher	Fire_M203_at_Location, Fire_M203_at_Target
M2A3 Main Gun, MG and TOW Launcher	fire_tow_at_target, deploy_weapon_below_speed
Mechanized HQ Section Script Enable	set-assigned-transport, set-dismounted-hq-section, set-mounted-hq-section, dismount-hq-section, unit-mount-subordinate-vehicle-teams
Mechanized Platoon Script Enable	set-dismounted, set-mounted-platoon, dismount-platoon, plt-assign-transport-defaults, unit-mount-subordinate-vehicle-teams

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Mechanized Squad Script Enable	set-assigned-transport, set-dismounted, set-mounted-squad, dismount-squad, suspend-crew, squad-assault, unit-mount-subordinate-vehicle-teams
Mechanized Team Script Enable	set-assigned-transport, set-dismounted, set-mounted-team, dismount-team, team-bound-in-lanes, team-assault, fire_team_attack_by_fire
Guided Missile Dynamics	set_freeze_fuel_levels, set_freeze_movement
Mortar Vehicle Script Enable	mortar-fire-at-location
Motorized Squad Script Enable	set-assigned-transport, set-dismounted, set-mounted-squad, dismount-squad, suspend-crew, squad-assault, unit-mount-subordinate-vehicle-teams
Multi-point Refueling System	set_cargo, establish_refueling_station, Deploy_Refueling_Boom, Stow_Refueling_Boom, stow_port_refueling_drogue, stow_starboard_refueling_drogue, deploy_port_refueling_drogue, deploy_starboard_refueling_drogue
Munition UAV (Rotary Wing)	munition-loiter-attack-rw
Munition UAV	orbit_object, set_freeze_fuel_levels, set_freeze_movement, munition-loiter-attack, clear-user-target
Naval Depth Charge Deployment	Drop_Naval_Depth_Charge, Drop_Naval_Depth_Charge_At_Location
Naval Mine Deployment	Drop_Naval_Mine, Drop_Naval_Mines_Along_Route
Naval Mine Dynamics	set_freeze_fuel_levels, set_freeze_movement
Naval Mine Sweep	Naval_Mine_Sweep
Nodong Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Parabolic Trajectory	set_freeze_fuel_levels, set_freeze_movement
Pedestrian Crowd Movement	set_freeze_movement, Crowd_Around, Crowd_Around_Location, Crowd_In_Front_Of, Crowd_Along_Line, Disperse_Crowd, Wander_In_Area, Protest, Protest_Along_Line, disaggregated_flee, set_navigation_priority, crowd_around_waypoints, Riot_In_Area, circle_around_location
Pedestrian Traffic Generator	Create_Initial_Pedestrians, Create_Pedestrians, Delete_Created_Pedestrians
Periscope	Raise_Periscope, Lower_Periscope
Request Fire Support	request_fire_support

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Rope Ladder	Rope_Ladder_Embarkation, Rope_Ladder_Disembarkation, Deploy_Rope_Ladders
Rotary Wing Attack	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, fly_to_position_on_circle, rotary-wing-move-to
Rotary Wing Cargo Operations	rw-retrieve-cargo, rw-unload-cargo
Rotary-wing Casualty Evacuation	rw-casualty-evacuation
Cargo	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, fly_to_position_on_circle, rotary-wing-move-to
Quadcopter Heavy Lift	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, fly_to_position_on_circle, rotary-wing-move-to
Quadcopter	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, fly_to_position_on_circle, rotary-wing-move-to
Dipping SONAR Sensor	Sonar_Dip
Utility	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, fly_to_position_on_circle, rotary-wing-move-to
VTUAV	fly_loiter_pattern_rwa, fly_circular_arc, orbit_object, fly_to_position_and_heading, sar_expanding_square, sar_parallel_pattern, sar_creep_pattern, fly_figure8, fly_cloverleaf, aircrafthold, aircrafthold_waypoint, set_freeze_fuel_levels, set_freeze_movement, fly_to_position_on_circle, rotary-wing-move-to
Scud C Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Scud-ER Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object

Table 81. Systems and configured scripted tasks, EntityLevel.sms (continued)

System	Scripted Tasks
Scud B Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Small boat	set_freeze_fuel_levels, set_freeze_movement, surface-move-to
Bomb Dynamics	set_freeze_fuel_levels, set_freeze_movement
Sniff Bombs	sniff_bombs
Sonobuoy Deployer	Deploy_Sonobuoy, Deploy_Sonobuoys_Along_Route
Space Shuttle	set_freeze_fuel_levels, set_freeze_movement
SS-25 Sickle Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Subsurface	set_freeze_fuel_levels, set_freeze_movement, subsurface-move-to
Surface Disaggregated Movement	set_freeze_fuel_levels, set_freeze_movement, surface-unit-move-to
Large Ship	set_freeze_fuel_levels, set_freeze_movement, surface-move-to
Tactical Radar Jammer	automatic_jamming
Tactical Data Link Air Message Generator	Set_TDL_Messaging
Tactical Data Link Ground Message Generator	Set_TDL_Messaging
Tactical Data Link Subsurface Message Generator	Set_TDL_Messaging
Tactical Data Link Surface Message Generator	Set_TDL_Messaging
UGM-133 Strategic Missile Launcher	launch-ballistic-missile-at-location, launch-ballistic-missile-at-target-object
Aggregate Artillery Call for Fire	unit_artillery_provide_indirect_fire, cease_fire, unit_artillery_ indirect_fire_at_location
Vehicle Platoon Script Enable	vehicle-platoon-marshal-on-route, unit-set-road-driving-options
Wing Tip Drogue Air Refueling Supplier	set_cargo, establish_refueling_station, Deploy_Refueling_Boom, Stow_Refueling_Boom, stow_port_refueling_drogue, stow_ starboard_refueling_drogue, deploy_port_refueling_drogue, deploy_starboard_refueling_drogue

E.3. Aggregate-level systems and configured scripted tasks

Table 82. Systems and configured scripted tasks, AggregateLevelBase.sms

System	Scripted Tasks
Mortar, 120mm (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
Aerial Refueling Boom (AggregateLevel)	refueling_mission
Aerial Refueling Drogue (AggregateLevel)	refueling_mission
122mm Howitzer (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
152mm Howitzer Auto (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
155mm Howitzer (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
Bomb, 2000lb JDAM (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Mk 45 5-inch Gun (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
9K330 Surface-to-Air Missile Launcher (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
9P140 MRL (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
AIM-120 AMRAAM Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, react_to_air_threats_air, air-to-air_attack
Communications Jammer (AggregateLevel)	EW_Attack, Activate_Jammer, Deactivate_Jammer
Bomb, GBU-24 (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Harpoon Antiship Missile (AggregateLevel)	attack_with_ship-attack_missile_flyout, Limited_Munition_Attack, Attack_with_Antiship_Missile, React_To_Naval_Threats_Missile
Bomb, KAB-1500L (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base

Table 82. Systems and configured scripted tasks, AggregateLevelBase.sms (continued)

System	Scripted Tasks
Kh-29-AS-14 Air-to-Surface Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_Antiship_Missile, Attack_with_Land-Attack_Missile
Kh-31 AS-17 Antiship Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_Antiship_Missile, React_To_Naval_Threats_Missile
M993 MLRS (AggregateLevel)	Indirect_Fire, Limited_Munition_Attack
Torpedo, Mark 54 (AggregateLevel)	Limited_Munition_Attack, Attack_with_Torpedo, React_To_Naval_Threats_Torpedo
Maverick Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_Antiship_Missile, Attack_with_Land-Attack_Missile
Patriot Anti-Air Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
R-60 AA-8 Anti-Air Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, react_to_air_threats_air
R-73 AA-11 Anti-Air Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, react_to_air_threats_air
R-77 AA-12 Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, react_to_air_threats_air
Radar Jammer (AggregateLevel)	EW_Attack, Activate_Jammer, Deactivate_Jammer
Bomb, RBK-500 (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Guided Bomb (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
RIM-66 SAM (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
Sidewinder Anti-Air Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, react_to_air_threats_air
Stinger Anti-Air Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_AntiAir_Missile, Automatic_Air_Defense, React_To_Air_Threats
Tomahawk Cruise Missile (AggregateLevel)	Limited_Munition_Attack, Attack_with_Land-Attack_Missile, attack_with_land-attack_missile_flyout

Table 82. Systems and configured scripted tasks, AggregateLevelBase.sms (continued)

System	Scripted Tasks
Unguided Bomb (AggregateLevel)	Bomb_Location, Limited_Munition_Attack, Ground_Attack, attack_air_base
Torpedo, Type 40 (RPK-6 Vodopad SUBROC) (AggregateLevel)	Limited_Munition_Attack, Attack_with_Torpedo, React_To_ Naval_Threats_Torpedo
Air-Drop Supply (AggregateLevel)	Provide_Supplies, Change_Supplying_Command
Aircraft Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, wait_for_escort, escort_air_unit, land_at_air_base
Biological Weapon (AggregateLevel)	Biological_Attack, NBC_Attack
Booby Trap Setter (AggregateLevel)	Create_Booby_Trap, Improve_Booby_Trap
Boom Aerial Refueling Receiver (AggregateLevel)	refuel_from_tanker
Bridge Builder (AggregateLevel)	Construct_Bridge, Improve_Bridge
Chemical Weapon (AggregateLevel)	Chemical_Attack, NBC_Attack
Ditch Digger (AggregateLevel)	Create_Anti-Tank_Ditch, Improve_Ditch, Create_Dry_Gap
Drogue Aerial Refueling Receiver (AggregateLevel)	refuel_from_tanker
Engineering Obstacle Destroyer (AggregateLevel)	Destroy_Obstacle, Destroy_Fortification, Destroy_Ditch, Destroy_ Bridge, Destroy_Explosive, Breach_Obstacles, Improve_Breach
Explosive Hazard and Simple Obstacle Destroyer (AggregateLevel)	Destroy_Explosive_Hazard, Destroy_Obstacle, Breach_Obstacles, Improve_Breach
FASCAM (AggregateLevel)	FASCAM
Fortification Builder (AggregateLevel)	Construct_Barricade, Improve_Fortification, Construct_Fortified_ Line, Construct_Fortified_Area, Construct_Strong_Point

Table 82. Systems and configured scripted tasks, AggregateLevelBase.sms (continued)

System	Scripted Tasks
Fuel Tank (AggregateLevel)	drop_fuel_tanks
Ground Combat Behaviors (AggregateLevel)	Attack_Target
Infantry Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed, React_to_indirect_fire
Mechanized Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed, React_to_indirect_fire
Mine Dispenser (AggregateLevel)	Create_Minefield
MOPP Gear (AggregateLevel)	Change_MOPP_Level
Motorized Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed, React_to_indirect_fire
Nuclear Weapon (AggregateLevel)	Nuclear_Attack, NBC_Attack
Obstacle Builder (AggregateLevel)	Construct_Abatis, Improve_Obstacle, Construct_Barbed_Wire, Construct_Berm
Ground Resupply (AggregateLevel)	Provide_Supplies, Change_Supplying_Command
Riverine Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed
Ship Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed
Simple Obstacle Destroyer (AggregateLevel)	Destroy_Obstacle, Breach_Obstacles, Improve_Breach
Subsurface Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, Flee_When_Routed
Tank Aggregated Movement (AggregateLevel)	Maximum_Speed_Modifier, Halt_Movement, Move_To_Location_ Plan_Path, Move_To_Waypoint_Plan_Path, halt_movement_ before_obstacles, Flee_When_Routed, React_to_indirect_fire

Table 82. Systems and configured scripted tasks, AggregateLevelBase.sms (continued)

System	Scripted Tasks
Tactical Data Link Air Message Generator (AggregateLevel)	Set_TDL_Messaging
Tactical Data Link Ground Message Generator (AggregateLevel)	Set_TDL_Messaging
Tactical Data Link Subsurface Message Generator (AggregateLevel)	Set_TDL_Messaging
Tactical Data Link Surface Message Generator (AggregateLevel)	Set_TDL_Messaging



MAK Products Glossary

This glossary defines terms used in MAK ONE product documentation.

A

absolute dead reckoning

A method of dead-reckoning in which an application uses the timestamps contained within state updates if the sender is using absolute timestamping. Contrast with relative dead reckoning.

aggregate

The combination of individual entities, aggregates, or both into a single object — for example, an organizational group such as a platoon.

API

Application Programming Interface.

articulated part

A part of an entity that is capable of movement relative to the entity, such as a wheel, turret, or gun.

attached part

An object that is attached to another object, such as a rocket attached to a jet.

B

body coordinate frame

A coordinate framework centered on an entity. To represent the orientation of an entity, an application uses Euler angles or a rotation matrix that rotates from a reference frame to the body coordinate frame.

C

channel

A channel renders the view of an observer in a 3D visualization application.

clipping

A feature that prevents the display of terrain and objects or parts of objects, that are closer than or farther than specific distances from the observer.

clipping planes

The range of distances from the observer (near clipping and far clipping) in which an application clips (does not display) objects.

culling

The process of discarding database objects which are not within view, and therefore need not be rendered and displayed.

Cartesian coordinates

A system for indicating location by means of three planes intersecting at right angles to one another at the origin.

D

Data Distribution Management

The set of High-Level Architecture services used to specify the routing of data between publishers and subscribers. DDM adds greater control over the Declaration Management (DM) services that only specify filtering based on the class of the data (e.g., ground vehicle versus aircraft). DDM uses the data content to express regions of interest and regions of affect whose overlap establish a communication channel. An example is geographic filtering where a publisher indicates a region around the entity or effect and a subscriber indicates a region expressing its sensor range.

dead-reckoning

A process by which an application calculates the expected location of an entity during periods between state updates, based on velocity, acceleration, and rotational velocity. See also absolute dead reckoning and relative dead reckoning.

DDM

See Data Distribution Management.

Defense Modeling and Simulation Office

The former name of the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Renamed Modeling and Simulation Coordination Office. Currently the MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness.

DIS

See Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

An object in an application that can render 3D data.

Distributed Interactive Simulation

The DIS protocol is a set of standards that govern how participating applications share information about the virtual world. The protocol specifies a set of packets, called protocol data units (PDUs), that communicate this information. Each PDU identifies the sender and contains other information depending on the PDU type. The DIS protocol also specifies when and how frequently PDUs are sent.

The DIS protocol has been superseded by the High-Level Architecture for use by the Department of Defense.

DMSO

See Defense Modeling and Simulation Office.

E

element definition

In VR-Forces and VR-Vantage, an element definition specifies all of the visualizers that control the various aspects of visualizing a scene element. For example, an element definition includes the main model or military symbology, trailing effects, cockpit displays, and so on.

emitter

A simulated device on an entity that emits electromagnetic radiation, for example, radar.

entity

An element in a simulation, such as a vehicle or a person, that is represented in the simulation through the issuance of state messages.

entity maintenance

A process by which the MAK Data Logger compensates for discontinuities following a time jump. The Logger sends out interim state messages for entities that were present before the time jump so that they do not time out before the next update message arrives.

entity-type

A list of seven components as specified by the DIS protocol and the HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific, narrowly-defined entity. If some components contain a wildcard, then the entity type refers to a class of entities. A larger number of wildcards indicates a broader, more general class.

The components of an entity-type are:

- Entity kind
- Domain
- Country
- Category
- Subcategory
- Specific
- Extra.

environmental process

According to the IEEE 1278.1a specification (DIS), environmental process PDUs communicate simple environment variables, small scale environmental updates, and embedded processes.

Euler angles

A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). These angles specify successive rotations needed to transform from a reference coordinate system to the entity's body coordinate system.

event

An interaction between objects or between an object and the terrain, such as firing of a munition, or a collision of entities.

exercise

The DIS term for a one or more interacting simulation applications. Compare to federation execution in HLA.

exercise connection

The object (DtExerciseConn) through which a VR-Link application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection. (All MAK ONE products are VR-Link applications.)

exercise ID

A numeric identification for a DIS simulation exercise.

F

FED file

Federation Execution Data file. The Federation Execution Data is the subset of FOM data needed and read by the MAK Products Glossary. VR-Link applications also read the FED file.

federate

A connection to the MAK Products Glossary. Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same federation execution.

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Controls the perspective of the observer.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed.

filter range

A setting that prevents distant entities from being processed while allowing distant terrain to appear normally.

FOM

See Federation Object Model.

FOM Module

In HLA Evolved and HLA 4, defines additional modular data content for a federation execution, usually extensions to an existing Federation Object Model. (Also supported by the HLA 1.3 and HLA 1516 versions of the MAK RTI. Per the HLA Evolved interface specification:

"A partial FOM (containing some or all OMT tables) that specifies a modular component of a FOM. A FOM module may contain classes not inherent to it but upon which the FOM module depends, i.e., superclasses to the modular components. These superclasses will be included in the FOM module either as complete or scaffolding definitions."

frame rate

The rate at which the application displays updated images.

G

ground clamping

A process by which the a 3D visualization application keeps an entity anchored to the surface of its terrain database, regardless of the altitude data contained in its entity state message.

geocentric coordinates

A coordinate system calculated with respect to the earth's center. The origin of the geocentric coordinate system is the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the north pole.

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MAK ONE applications, geodetic coordinates consist of latitude and longitude in radians, and altitude in meters above the reference ellipsoid.

gridded data

Data that has been processed in a rectangular array of points, in X, Y or latitude/longitude, at which single data values define a two dimensional function. According to the IEEE 1278.1a specification, gridded data transmits information about large-scale or high-fidelity spatially and temporally varying ambient fields and about environmental processes and features.

guise

An alternative entity type used to display an object depending on the force ID. For example, a tank could look like an M1A1 to friendly forces and a T72 to hostile forces.

H

head-up display

A set of indicators and readouts superimposed onto a graphics display. Also called an overlay.

heartbeat

In DIS, the frequency with which current Protocol Data Units are sent to the network regardless of whether the entity's state has changed.

High-Level Architecture

The High Level Architecture (HLA) for simulations is a U.S. Department of Defense (DOD)-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the DOD.

HLA

See High-Level Architecture.

I

IG

image generator

interaction

A message describing a simulation event. An interaction describes an event, it does not update an objects state.

L

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MAK Data Logger remotely.

logical range

A suite of TENA Resources, sharing a common object model, that work together for a given range event. Similar in concept to the HLA Federation.

Local RTI Component

The libraries on a computer that implement a Run-Time Infrastructure. A federate communicates with the HLA network through the LRC.

LRC

See Local RTI Component.

M

MÄK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MAK ONE products.

message

A general term used to refer to DIS PDUs and HLA interactions and state updates. In TENA, a message is similar to an HLA interaction.

MIM

The MOM & Initialization Module (MIM) allows for extensions to be made to the High-Level Architecture standard MOM. It is a subset of the Federation Object Model that contains OMT tables that describe the HLA MOM. The MIM also contains additional predefined HLA constructs such as object and interaction roots, data types, transportation types and dimensions. HLA specifies a standard MIM that is incorporated into all FDDs automatically by the Run-Time Infrastructure. The standard MIM can be replaced with a user-supplied MIM containing the standard MIM plus extensions.

model definition

In VR-Forces and VR-Vantage, a model definition specifies the 3D model and other basic attributes of a model. It is referenced by an element definition.

Modeling and Simulation Coordination Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. The MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness. (Formerly DMSO)

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MÄK Technologies Lisp.

N

NETN FOM

The NETN FOM (NATO Education and Training Network Distributed Simulation Federation Object Model) is an extension of the RPR FOM 2.0, allowing other federates using pure RPR FOM to interoperate with applications using the NETN FOM.

O

orientation clamping

Adjusts the pitch and roll of an entity so that it appears properly seated when the terrain is inclined. For example, if a tank is moving horizontally across the face of a hill, orientation clamping prevents the tank from appearing level, and therefore, partially embedded into the hillside. Used with ground clamping.

object

An element in a simulation that has persistence, as opposed to an interaction, which is a transient element.

object handle

An integer that an application uses to identify a particular object in Run-Time Infrastructure service calls. An object handle is meaningful only to a particular federate. The same object can be known to different federates by different object handles.

object name

A character string that can be used to identify an object. In HLA, the object name is known to the Run-Time Infrastructure, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

observer

In MAK ONE 3D applications, the point of view into the scene. (Called the eyepoint in MAK Stealth 6.x and earlier.)

P

PDU

See Protocol Data Unit.

Protocol Data Unit

A unit of data message (packet) that is passed on a network between DIS simulation applications.

R

radio transmitter

A simulated device on an entity, capable of transmitting radio communications.

radio receiver

A simulated device on an entity, capable of receiving radio communications.

Real-time Platform Reference FOM

An HLA reference Federation Object Model based on the DIS protocol.

recording

A MAK Data Logger file that stores a history of the interactions in a simulation for playback.

reflected entity list

A list of entities simulated by remote applications (federates in HLA) and about which the local simulation has received information over the network.

reference FOM

A Federation Object Model designed to be used as a whole, or with modification, by a wide variety of similar federation executions.

relative dead reckoning

A method of dead reckoning in which an application approximates the location of an object based on the local time of receipt of the last state update message. Contrast with absolute dead reckoning.

remote display engine

A display engine running in an application that is separate from the master application, and which is controlled by the master application.

RID file

See RTI Initialization Data.

RTI Initialization Data

Data required by a Run-Time Infrastructure during initialization, independent of the FOM being used. RID data is usually dependent on a specific implementation of the RTI.

RPR FOM

See Real-time Platform Reference FOM.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

A library and other supporting software that implements the HLA interface specification. All federates communicate with one another in an HLA environment through RTI functions.

S

scene element

In VR-Vantage and VR-Forces, any object that is rendered in the scene, such as an entity model, interaction, or tactical graphic.

Simulation Interoperability Standards Organization

A group that seeks to promote modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users, world-wide.

simulation time

In VR-Forces, simulation time is used in dead-reckoning of remote entities and thresholding of local entities. Typically, simulation time is set once during each iteration of the application's main simulation loop so that all entities are dead-reckoned based on the same value of current time.

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position are not so abrupt as to be visually disconcerting.

state

The current status of an object, including location, direction of movement, extent of damage, and so on.

T

tape

An alternative term for a Logger recording. Not a physical magnetic tape.

terrain following

In a 3D visualization application, causes the observer (eyepoint) to maintain a constant distance above the terrain surface. The observer's height changes in tandem with the peaks and valleys as it passes over the geography.

timeout

The period of time in which an application continues to display an entity after that entity's update messages have stopped appearing on the network. Time-outs are usually not used in HLA because there is no set frequency (no heartbeat) for transmitting messages.

timescale

A factor by which time is accelerated or slowed during playback of a Data Logger recording.

topographic coordinates

A right-handed Cartesian coordinate system whose X-Y plane is tangent to the earth's surface at the origin, with the positive X axis pointing north, the positive Y axis pointing east, and the positive Z axis pointing down. There are an infinite number of topographic coordinate systems – one for each point on the earth's surface.

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

A method used by to smooth positional discontinuities that could occur when new state updates arrive.

U

UDP port

A network channel through which an application sends and receives data for DIS exercises.

Universal Transverse Mercator

In general, a non-cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth.

UTM

See Universal Transverse Mercator.

V

view control messages

A set of programmatic messages that let you control the view in VR-Vantage applications remotely.



Index

-

-1 (wildcard), 1471

#

2D
  icon, 381
    changing, 1348
    changing size, 1080
    customizing color, 479
    customizing transparency, 479
    label, 470
    scale, 1080
    using image for, 1349
  label
    background, 476
    text color, 474
    text shadow, 476
  observer
    attachment behavior, 1220
    coordinate system, 1182
  projection, change to 3D, 1164
  window, 137
2D Icon model set, 1172
2D View Layer
  loading saved view, 1206
  saving view, 1206
  selecting, 1262
3D
  colorized model set, color, 479
  label
    background, 476
    text color, 474
    text shadow, 476
  model, 381
    changing, 1347
    DI-Guy, 654
    spot report, 295
  model licensing, 92
  observer coordinate systems, 1182
  projection, change to 2D, 1164
  settings, 217
  window, 137
3D Cartesian coordinate system, 802
3D Colorized Models model set, 1172
3D Models model set, 1172
3DS
  ASCII Export, 1401
  ASE, 1401
  ASE file, 1399

A

AAR, See after action review
Abandon Plan command, 982
abandoning plan, 982
  confirmation prompt, 561
  independent task, 554
abatis, constructing, 720
Absolute observer mode, 1214-1215
absolute, pathname, 355
acceleration
  displaying, 470
  specifying measurement unit, 1634
acceleration-factor parameter, 507
acceleration-to-rpm-factor parameter, 311

accumulation
rain, 329
snow, 329
ACO, 245
action
category, 581, 750, 765
action table, 1465
actionCategories.tsk, 581, 766
Activate Jammer task, 737
activating
jammer, 737
scenario event, 283
scenario event from plan, 284
activation
indirect fire, 1035
spawn points, 418
Activation set data request, 1038
active
sonar, 310
stereo, 1616
tactical graphic, 1035, 1038
Active Sonar Mode set data request, 866
active-radar-sensor system, 540
activity
adding, 1429
aggregate, 915
Activity set data request, 1429
actuator, 366-367
indirect fire, 1534
missile-collision, 1533-1534
missile-maneuver, 1533-1534
warhead, 1536
warhead detonation, 1536
adaptive tick rate, 217
Add Jamming Target set data request, 866
adding
articulated part to simulation object, 1387
channel, 1601
DI-Guy animation, 1465
DI-Guy character, 1464
embarkation slot, 1442, 1444, 1451
entities to crowd or herd, 652
feature layer, 1244
global commands to plan, 969
key mapping, 1651
media to scenario event, 279
model, visual, 1351
object geometry, 1380
observer, 1167-1168
state properties, 1490
targets to priority list, 905-906
text to title bar, 1633
vertex, 1030
additional-algorithm-config parameter, 1561
additionalDestinationAddresses parameter, 1678
additional-opds parameter, 354
additionalSystemScriptsDirectory parameter, 1668
addMulticastAddr parameter, 1678
address
host, 181
address, street
move observer to, 1207
advisory message, 1678
advisory, ignoring, 175, 185, 1151
aerodrome, 1292
base, assigning, 872
centering on, 1294
create, 1292
in terrain, 1292
information dialog box, 1294
landing aircraft at, 596, 611
lighting, 1295
overview, 1292
promote, 1293
publish, 1292
runway, active, 1292
Aerodrome Light Conditions set data request, 1295
Aerodrome Undercarriage Inspection System set data request, 1295
Aeronautical Charts terrain, 1239
after action review
3D view, 76
API support, 88
batch mode, 273
MAK Data Logger, 274
agent, contamination, 532
aggregate as
specifying unit option, 1406
units that do not allow, 1407
aggregate-comms-jammer system, 540
aggregate-formation-controller parameter, 1566
aggregate-level
activity, 915
attack by fire task, 714
combat, 532
entity, configuring, 1418
modeling, 368
movement
air, 550-551
land, 550-551
terrain, 550
unit, group, 551
unit, individual, 550
watercraft, 551
SMS, 368, 528

warfare model, 528, 1418
aggregate-level unit parameter, 1419
aggregate-object-param parameter, 1477, 1501
aggregate-param parameter, 1688
aggregate-radar-jammer system, 541
AggregateTacticalLevel.sms, 368
aggregating objects in Order of Battle, 493
aggregation
units that do not allow, 1407
agility, terrain, 84
AI, 123
AI enabled, 866
aim
sensor, 899
Aiming set data request, 868
air
humidity, 333
pressure settings, 327
tactical navigation, 904
temperature, 333
temperature settings, 327
turbulence, 340
air base
assigning aircraft, 872, 916
attacking, 713
aviation fuel supply, setting, 923
facilities, setting, 919
landing at, 734
munition supply, setting, 923
personnel, setting, 923
air corridor, 1016
air crew, setting for air base, 923
air defense, 716
air unit
attacking air base, 713
combat, 1437
fuel tank, dropping, 738
airbase, See air base
aircraft
assignment
deleting, 917
editing, 916-917
attacking, 625
attacking with gun, 625
base aerodrome, 872
crank maneuver, 587
land at air base, 611
land at airport, 596
land on route, 596
land on runway, 597
landing at air base, 734
lighting, 868
refuel, 587, 610, 738
screen display, 1598
speed, 464, 473
strafing target, 628
takeoff from runway, 619-620
tanker
lighting, 904
refuel self, 692
Aircraft Clutter Area, 428
Aircraft Lighting set data request, 868
airdrop, 608
Airdrop Along Route task, 585
Airdrop from Aircraft embarkation task, 684
Airdrop task, 683
airlift cargo, 611-612
airport, See also aerodrome
airspace, 1016
airspace control order, 245
Air-to-Air Attack task, 713
Air-to-Ground Air Base Attack task, 713
Air-to-Ground Attack task, 714
AIS Data set data request, 867
Ala Moana terrain, 1238
Allow Crossing set data request, 1039
all-tasks-exclusive parameter, 580
alpha to coverage, 207
VRV_ALPHA_TO_COVERAGE_RENDERBIN environment variable, 208
altitude, 470, 801
change from 2D to 3D, 1164
entity
displaying, 480
specifying, 387
fixed-wing entity, 510
flying to, 589, 591
measurement unit, 1634
ordering entity to, 605
route
displaying, 1001
setting, 388, 1030
setting, 386, 388, 869
fixed-wing entity, 510
in dialog box, 387
manually, 387
simulation object, 393
pasting, 398
symbol decoration, 475
testing entity for, 939
track history, displaying, 1056
vertex, 1000
changing, 1031
Altitude condition, 939
Altitude set data request, 510, 869
Always Join Session with Open Database, 141

Always Join Session with Session Database, 141
ambient
  air temperature, 327, 333
  color, 1089
  value, material, 1089
ambient occlusion
  enabling, 209
  map, 86
  settings, 211
ammo select file, 1527
ammo select table, countermeasure, 311
ammunition, 537
  amount, 915
  configure, 1529
  creating, 1394
  freeze levels, 881
  pacing and tracking, 915
  specifying amount, 896
  status, 465
  unit, 1420, 1422
Ammunition Pacing/Tracking set data request, 915
Ammunition set data request, 915
anaglyphic stereo, 1616
  configuring, 1617
angle, damage, 1551
angle-of-incidence parameter, 1551
angleSegmentMax parameter, 1139
animal herd
  adding members, 652
  area creation, 434
  disallows aggregation, 1407
  unit creation, 433
animated movement
  configuring, 1399
  movement file, configuring, 1403
Animated Movement task, 585
animation
  attack, 1059
  DI-Guy, 180, 654
    adding, 1465
    choosing, 1466
    embarked, 1447
  entity, 585
  fast roping, 686
  ladder extraction, 688, 690
  ladder insertion, 690-691
  observer view transition, 1204
  rope extraction, 686-687
  rope insertion, 688
  testing DI-Guy, 939
  urgency, 908
Answer Questions button, 805
anti-air
  missile, 715
  resource, 537
  weapon, 716
anti-aliasing, 164, 207, 1661
anti-personnel, 537
anti-ship
  missile, 715
  resource, 537
Anti-ship (Fixed Depth) task, 641
anti-submarine missile, 638
anti-tank
  ditch
    construct, 721
    destroy, 726
    digging, 721
    improve, 728
  resource, 537
Any, as condition parameter, 963
apparent signature, 1510, 1516
appearance
  bit, 1090
  DI-Guy, 180
    random, 1385
  pasting, 397
  power plant, 310
  specifying, 869
  specifying for DI-Guy model, 1383, 1386
  testing DI-Guy, 940
Appearance set data request, 869, 1090
appIdRange parameter, 1678
application
  data, specifying location of, 164, 178, 187, 1306
    Simulation Object Editor, 1306
  ID, 115
  number, 156, 175-176, 178, 189, 274, 357, 1149, 1668
    configuring, 1668
    default, 189
    range, 178
    specifying at startup, 189
  observer, visibility, 1172
  priority, 168
  remote
    notifying of VR-Forces run and pause, 273
    observer, 1173, 1207
application ID, 176
  range, 176, 186
application, MAK ONE
  build information, 96
appNumber parameter, 1668
arc
  drawing, 1014

segment, 1138
architecture component, 366
area, 994, 997
air turbulence, 340
clutter, 428-429
promote entity, 893
drawing, 1012
editing, 1033
extruded, 940, 1016
feature list, 1258
fill style, 1033
fortified
construct, 724
destroy, 727
improve, 729
hazard, 532
high fidelity damage, 1561
minefield, 1021
navigation
creating, 1277
importing, 1287
NBC, 532
pedestrian, 436
sector, 1014
suppressive fire at, 520, 637, 645
types, 1012
Arm Mine at Depth task, 624
Armed set data request, 870
arming
IED, 870
naval mine, 624
armor-piercing power domain, 1547
arrestor gear, 1297
Arrivals Hall terrain, 1238
arrow
adding to line, 1033
freehand line, 1013
sensor contact line, 1065
Arrowhead Style set data request, 1039
articulated part, 1387
adding to simulation object, 1387
animating in SOE, 1389
attaching observer to, 1212
displaying, 1353
editing for simulation object, 1388
removing from simulation object, 1389
smoothing, 215
tasks for moving, 698
Articulated Part set data request, 870
articulated parts, 1078
artillery
Fire for Effect task, 634
unit, 718
ASCII Export file, 1399, 1401
ASE file, 1399, 1401
ASE format, 1401
Ask to Join Session, 141
Assault task, 675
assembly, 1420, 1424
adding to entity, 1427
copy, 1425
create, 1424
from unit, 1426
edit, 1425
remove, 1428
roll up rule, 1428
rolling up, 1428
update from unit, 1426
assertOnBlockingTerrainCalls parameter, 1247, 1668
Assign Aircraft set data request, 916
Assign Transport set data request, 909
assigning
echelon ID, 365
plan to multiple simulation objects, 980
session ID, 133
Assume Coil Formation task, 674
Assume Unknown is Hostile set data request, 870
asynchronous
I/O, 1679
enable, 177, 184, 193, 1150, 1679
atmospheric condition layers, 324
attach
3D modes, 1214
context, 1182
observer to articulated part, 1212
Attach Components to Remote Entities On parameter, 230
attach mode, 85, 1177, 1182, 1214
2D behavior, 1220
Absolute, 1215
Follow, 1215
Mimic, 1215
Mimic Location Track, 1217
Mimic Track, 1216
movement, attached context, 1182
prop, 1214
Space Follow, 1217
Tether, 1218
Tether Location Track, 1219
Tether Track, 1219
Track, 1219
Attached Near Clip Altitude attribute, 1607
Attached set data request, 871
attaching observer
to object using keyboard, 1211

to simulation object or prop, 1210
attachment
list, filtering, 1213
order, 1211
preservation of view changes, 1211
attack
air base, 713
air-to-ground
aggregate, 714
entity, 630
animation, 1059
anti-air missile, 715
anti-ship missile, 715
indicator, 1059
limited munition, 532
missile, 532
Attack Aircraft task, 625
Attack Aircraft with Guns task, 625
Attack By Fire in Rotation task, 676
Attack By Fire task, 676, 714
attack parameter, 1435
Attack with Anti-Air Missile task, 715
Attack with Anti-Ship Missile task, 715
Attack with Land-Attack Missile task, 715
Attack with Torpedo task, 716
Attention Grabbing set data request, 871
attribute
Attached Near Clip Altitude, 1607
Dynamic Near Clip Attached Policy, 1607
Dynamic Water Visibility Enabled, 1611
Dynamic Water Visibility Maximum, 1611
Fixed Near Clip, 1607
Fixed Near Clip When Attached, 1607
itemname, 1640, 1646
Near Far Clip Policy, 1606
Reduce Z Fighting Ratio, 1606
sensor, 1604
setting simulation object, 869
window, 1603
attrition
rate, 528
unit, 535
audio, 1142
adding to scenario event, 279
global settings, 1142
MAK radio settings, 1158
auto connect period, 164
auto tag, applying, 1330
automatic
attachment, 1029
countermeasures, 311
embarkation, 409
reorganization, 895
running of scenarios, 269
Automatic Air Defense task, 716
Automatically Close Doors set data request, 872
automaticallyCreateGlobalEnvironment parameter, 1668
Autonomous Actions Enabled set data request, 866
autonomous lasing, 304
auto-reorganize parameter, 353
autoTagInformation.csv, 1330
aviation fuel
air base supply, 923
drop fuel tanks, 738
refueling, 587, 610
transfer, 692
Avoid Roads, 890
avoiding
collisions, 873, 1276
objects, 1568

B

back-end, 114, See also simulation engine
balancing load, 255
console, 1660, 1670
dropping from exercise, 118
feature representation, 1244
frame rate statistics, 200
heartbeat, 1675
late joining, 118
mapping objects to, 357
missing, 119
multiple, 114, 118, 357
reloading OPD, 1670
remapping, 119
in batch mode, 272
sending messages, 114
starting, 129
as daemon, 180
synchronizing, 142
background
activity, 428
aircraft, 428
label, 476
ship, 429
background process, 740, 746, 812
Bakersfield topo map terrain, 1238
balancing
object load, 255
ballistic gun, firing, 632
ballistic missile, 319, 639
firing, 320

movement file, configuring, 1403
movement, configuring, 1399
scripted movement, 1399
target event, 321-322
ballistic-missile-entity-state-repository, 1476
ballistic-missile-param parameter, 1476
barbed wire
construct, 721
destroy, 727
improve, 729
barricade
construct, 722
destroy, 727
improve, 729
Base set data request, 872
batch file, 269
command-line option, 179, 272
configuring, 1668
editing, 270
batch mode, 269-270
command line, 179, 272
remapping, 272
batch scenario, running, 272
batchScenarioFileName parameter, 1668
beacon
emergency, 878, 1391
wake, 345
beam, 894
emitter, 310, 878
standard and tracking, 878
bearing
bullseye, 1026
measurement unit, 1634
view with range line, 1130
behavior
scripted, 585
simulation object, 367
behavior set, 752
assigning to force, 580
creating, 578
script availability, 766-767
berm
construct, 722
destroy, 727
dynamic terrain, 1023
improve, 729
bin, render, 208
binary key mapping, 1651
binding, variable, system definition file, 1486
Biological Attack task, 716
biological contamination, 532, 716
bit depth, 164
and stencil, 207
bitmap, 1243
adding to scenario event, 279
location of, 103
block
damage-by-munition-power, 1551
damage-model, 1551
determinant, 1551
high-fidelity-terrain-damage-table, 1561
munition, 1546
power-list, 1547
standard-terrain-damage-table, 1560
blockOnAsynchronousOperations parameter, 1669
blowing dust, 337
blowing snow, 337
blur, 1116-1117
BMP, 1243
boat, See watercraft
body coordinates, 1177
bomb
creating, 1393
dropping, 646-647
laser guided, 646
Bomb Location task, 717
bomb-bay weapon system, 1536
bomb-entity-state-repository, 1476
bomb-param parameter, 1476
booby trap, 722
aggregate-level modeling, 535
destroying, 727
improving, 727
boolean operator, 942
boom, refueling, 700
border, icon, 1080
Bound In Lanes task, 673
bounding box, See bounding volume
bounding overwatch, 657
Bounding Overwatch Successive Continuous task, 673
bounding volume, 1352
displaying, 1353
selecting objects, 455
simulation objects, 481-482
specifying, 1377
unit, 448, 1414
color, 450
box
drawing, 1016
extruded, 1016
resizing, 1032
breach
fortification, 719
improving, 728

obstacle, 719
Breach Obstacles task, 719
bridge, 400, 1377
building, 723
destroying, 726
improving, 728
brightness, 1117
broadcast parameter, 293
broadcasting
state update, 1172
stealth position and orientation, 1173
Brooklyn terrain, 1238
buffer
depth, 164
buffer size, 177, 182, 1150
build information, 96
building
bridge, 723
conditional statement, 956
ditch, 723
dry gap, 723
fortified area, 724
building (structure), 400, 1377
creating entity in, 386
embedding in terrain, 401
going inside, 386
bullseye
Move to Bullseye Offset task, 605
range, 1026
spider web, 1028
tactical graphic, 1025
bump map, 86
buoy
wake, 345
buoyancy
enabling or disabling, 1111
LOD distance, 1111
burst
length, 421, 424
period, 421, 424
spawn pattern, 421, 424
button
Answer Questions, 805
common for model definitions, 1630
Joystick, 1582
toolbar, 1646
toolbars, 135
buttonGroup property, 857

C

C++
scripted task, 770
task, 745, 795
cache
clearing model instancing, 204
model instance, 204
specifying directory, 164
caching
terrain data, 90
calculating damage, 1546
calculating support point, 1379
California terrain, 1238
Callsign set data request, 873
callstack.log file, 98
camera
effects, 1116
jitter, 1198
moving, 1186
position and orientation, 1615
shaking, 1198
ways to move, 1185
Camera Orientation Offset, 1604
Camera Position Offset, 1604, 1615
CameraFX, 1164
camouflage
marking, 1357
can-be-radar-jammed parameter, 540
canceling
generation of navigation data, 1284
reactive task, 576
can-pivot parameter, 622
Capabilities set data request, 873
capability, specifying, 873
capturing screen image, 138
car
getting in and out of, 683
hiding a driver, 882
car bomb
arming, 870
detonating, 875
Car Door task, 698
car door, opening and closing, 683
car, displaying drivers, 1085
cargo
carrying, 611
embarking, 612
loading, 612
unloading, 612
cargo door
closing, 700
opening, 699
carrier
embarking entity on, 406
placing entity on, 386
carrier air wing, disallows aggregation, 1407

Cartesian coordinate system, 802-803
CAS, See close air support
Categories, clearing list, 1342
category
  action, 750, 765
  echelon, 364
  undoing changes, 1329
  weapon and ammunition
    adding, 1440
CDB
  performance, 165
Cease Fire task, 625
CEO. See combat engineering object, 533
certainty level for spot reports, 294
cgfDispatcherReceivePort parameter, 1669
chaff, 640
  launching, 311
Change MOPP Level task, 732
Change Posture task, 733
changing
  altitude of vertex, 1031
  channel attributes, 1604
  color, tactical graphics, 1033
  colorized model, 1347
  control object, 1029
  entity
    enumeration, 1338
    name, 393
    parameter, 1380
    tag, 1337
  extended label index, 478
  field of view, 1609
  force, 879
  force hostility, 300
  frame rate mode, 354
  line direction, 1033
  marking, 888
  model set, 1353
  name
    entity, 393
    tactical graphic, 393
  object tag, 1337
  order of objects in OOB, 495
  posture, 733, 924
  script ID, 753
  simulation object name, 1340
  Simulation Time Scale Toolbar range, 261
  site ID, 357
  spot report icon, 1351
  subordinate order, 1410
  tactical graphic
    color, 1033
    name, 393
  terrain database for scenario, 356
  width of line, 1033
changing vertex altitude, 1031
channel
  attribute
    sensor, 1604
  changing attributes, 1604
  keywords, 1604, 1613
    compass, 1180
  multiple, 1613
  new, 1601
  observer, 1167
  output, enable, 181
  projection resize policy, 1607
  property, 1604
  removing, 1602
  seamless, 1661
  text chat, 1632
Channel Name, 1604
character
  DI-Guy, 180
  DI-Guy, adding, 1464
  specifying for DI-Guy model, 1383, 1386
character_animations_table.mtl, 1464
chat window, 1631
  channels, configuring, 1632
check() function, 789
checkInit() function, 789
checkpoint, 247, 264
  deleting, 267
  disabling periodic, 266
  rolling back to, 262
  saving, 265
  saving from snapshot, 265
  scenario, 265
  scheduling periodic, 266
  script, limitations, 793
Chemical Attack task, 717
chemical contamination, 532, 717
child SMS, See simulation model set, derived
choice, list in Qt Designer, 857
chop, wave, 341
choppiness
  ocean, 344
CID, See combat identification level
CID level, 302
  for firing at target, 304
  icon color, 303
circle
  around a point, 607
  drawing, 1013
  extruded, 1016
Circle Around Location task, 650

Civilian Idle task, 650
Civilian Pray task, 650
Civilian Protest task, 651
Civilian Talk task, 651
civilian visit point, 436
civilian, creating crowd, 432, 434
Clamp Near Clip When Magnified policy, 1606
clamping
  entities, 1248
  ground configuration, 1249
  human entities, 881
  orientation configuration, 1249
class
  DtCgfDispatcher, 358
  DtDetonationManager, 1562
  DtSimComponentManager, 1473
  DtSimObjectNetInterface, 1473
  DtSimObjectStateRepository, 1473
  DtSimTaskManager, 1473
  DtVrfObject, 1469, 1473
  DtVrfObjectBaseStateRepository, 1469
  DtVrfObjectParameters, 1469
  DtVrfObjectPlanManager, 1473
  DtVrfRadio, 1574
  DtVrfRemoteController, 358
  FeatureSet, 784
  Location3D, 801
  Lua, 784
  SimObject, 784
  Vector3D, 801
  VectorGeoc3D, 803
  VectorOffset3D, 803
  vrf, 785
class doc, location, 104
Classified CID level, 302
Clear task, 676
clearing
  default window layout, 1626
  model instancing cache, 204
  object console, 487
  snapshot list, 268
  target priority list, 906
  view list, 1204
click to create, 382
click to locate, 382, 384
climb rate, 591
clipboard, 485
clipping plane, adjusting when creating simulation objects, 386
clock mode, 122, 179, 354
clockMode parameter, 1669
close air support
  aggregate-level, 714
  entity-level, 630
  strafing, 628
Close Car Door task, 699
Close Cargo Door task, 700
Close Door task, 651
Close Window task, 651
closing
  all windows, 146
  application, 146
  car door, 683, 699
  cargo door, 700
  door in building, 651
  formation, 517
  simulation, 263
  simulation engine, 146
  window in building, 651
cloud
  cover, 324, 333
  disabling, 1100
  shadows, 1100
  skybox cube map, 1096
  smoke, 640
  synchronization, 168, 323, 342
clutter object
  aircraft, 428
  area, 428
  promoting, 893
  ship, 429
cockpit display, 1226
  enabling or disabling, 1225
  enabling with keyword, 1613
cockpitOnly keyword, 1613
code
  country, 1669
  laser, 885
    assigning, 304
coefficient probability, 1530
coefficients determinant, 1553
coil defense, 674
collapse unit, 444
collateral damage
  probability table, 1531
collision avoidance, 1276
  configuring, 1568
  specifying for entity, 873
Collision Avoidance Enabled set data request, 873
color
  2D icon, 479
  ambient, 1089
  compass, 1181
  diffuse, 1089
  draw call, 215

emitter volume
frequency, 1136
pulse rate, 1136
entity label, 469
fog, 328, 333
force, changing, 479, 1333
intervisibility, configuring, 1127
label background, 476
label text, 474
partially identified simulation objects, 303
radio communication line, 1076
sensor contact line, 1065
sensor field of view, 899
specular, 1089
squawk indicator, 1076
tactical graphics, 1001, 1033
terrain, 1260
unit bounding volume, 450
Color set data request, 1040
colorized model
changing, 1347
using set, 1172
combat, 528
aggregate-level, 532
power, 528
unit, graphics, 1059
combat engineering object, 533
configuration, 1439
creating, 535
new model, 1418
partial completion, 536
percent complete, specifying, 923
rate of creation, 536
status, 536
combat identification level, 302, 1510
Combat parameters, 1437
combined mode, vrfLauncher, 188
Come to Stop task, 586
command
Abandon Plan, 982
Create Sensor View, 978
Delete Self, 422
Disembark, 410
Embark On, 407
global, adding to plan, 969
Issue Plan, 973
menu, configuring, 1640
Restart Plan, 981
Send Text Message, 972
Skip Task, 572
Test Result, 979
command line
loading terrain, 174
options, 164
Simulation Object Editor, 1306
command panel, 558
icons, 560
show or hide, 558
command-line option, 163
--, 112, 175, 185, 1151, 1309
-A, 152, 176, 180, 192, 1149, 1678
-a, 156, 175-176, 178, 189, 1149, 1668
--additionalSystemScriptsDirectory, 178
--antiAliasing, 164, 207, 1661
--appCacheDir, 164
--appDataDir, 164, 178, 187, 1306
--appIdRange, 176, 186
--appIdRangeMax, 178
--appNumber, 156, 175-176, 178, 189, 1149, 1668
--attributePduStrategy, 179
--autoConnect, 164
-B, 179, 272
-c, 150, 173, 187-188
-C, 164, 187
--clockMode, 179
--config, 187
--connection, 150, 187-188
--connectionless, 179
--countryCodesMappingFile, 179
-d, 183
--daemon, 180
--dataDir, 164, 180, 187, 1306
--defaultObjectTimeout, 176, 186
--defaultSimulationModelSet, 164
--depthBuffer, 164
--destAddrString, 152, 176, 180, 192, 1149, 1678
--deviceAddress, 150-151, 177, 180
--diGuyAnimationsFile, 180
--diGuyCharacterDataFile, 180
--dis, 164
--dis5Timestamps, 180, 1149
--disableDynamicTerrain, 165
--disableErrorDialog, 165
--disableGlBufferTracking, 165
--disableIndirectEntities, 165
--disableIndirectExternalRenderer, 165
--disableKDTrees, 165
--disableModelSets, 165
--disableParallelPublisherTick, 180
--disableParallelTick, 180
--disableRemoteDiscovery, 175, 177, 180
--disableRequestUpdates, 175
--disableReverseZBuffer, 165
--disableTerrainObserverViews, 165
--discoveredModelPreloadFileName, 219
-disPort, 192

--disPort, 152, 177, 186, 192, 1151, 1679
--dispSetting, 165
--disVersion, 152, 177, 186, 193, 1151
--doNotCheckPluginVersions, 166
--doNotLoadVrfPlugins, 166, 180, 1306
--doNotUseConsole, 180, 183
-E, 167
-e, 167
--enableChannel, 181
--enableDofConstraints, 166
--enableLogFileTimestamps, 166
--enablePduProfiling, 181
--enablePersistentShadowTechnique, 166
--enableTracyProfiling, 166
enableVSync, 216
--enableVSync, 166
--entDispSetting, 166
--entityDefsDir, 166
--entityGridCellSize, 167
--entTypeMap, 167
--envSetting, 167
--exConnConfigFile, 167, 181, 1150
--execname, 151, 175, 185, 190, 1150, 1677
--exerciseId, 152, 177, 186, 193, 1151, 1679
--exitOnTestResultCommand, 181
--extraArgs, 167
-f, 175, 185, 191-192, 1151, 1677
--factoryRootDir, 167
--federateName, 175, 185, 1150, 1677
--federateType, 175, 185, 1150
--fedFileName, 151, 175, 185, 191-192, 1150, 1677
--fileNotifyLevel, 181, 1149
--fileTransporterReceivePort (DIS), 175, 177
--fomMapperInitData, 175, 185, 1150, 1677
--fomMapperLib, 175, 185, 191-192, 1151, 1677
--fomModules, 175, 185, 1151
--forceTextShaping, 167
--fowPerspective, 167
--frontEndPID, 181
--fullScreen, 168, 1629
--fullyLoadTerrain, 181
-G, 189
--gui, 168, 187, 1307, 1640
-h, 111, 175, 185, 188, 1151, 1309
-H, 175, 177
--help, 111, 175, 185, 188, 1151, 1309
--highestPriority, 168
--hla1516e, 168
--hostAddressString, 175, 177, 181
-i, 133, 150, 152, 176-177, 183
--ignore_rest, 112, 175, 185, 1151, 1309
--ignoreAdvisories, 175, 185, 1151
--ignoreZoomForSpeedtreeLod, 168
-K, 165
-L, 182, 251
--limitEarthSplattingTextures, 168
--limitEarthTerrainTextures, 169
--limitEarthVegetationTextures, 169
--limitEntityTextures, 169, 1307
--limitEntityTexturesExclusions, 169, 1307
--limitGpuMemoryPercentage, 170, 1307
--limitTerrainTextures, 169
--loadPlugin, 190
--locale, 112, 170, 187, 189, 1306
--localSettingsDesignator, 176, 186, 1151
--logFileName, 170, 181, 1149
--mcastTtl, 177, 182, 1149, 1679
--mergeEntityTypeMap, 170
--mergeHierarchyEntityDef, 171
--mergeHierarchySpotReportDef, 171
--mergeHierarchyTacticalGraphicsDef, 171
--mergeHierarchyUnitDef, 171
--mergeLeafEntityDef, 171
--mergeLeafSpotReportDef, 171
--mergeLeafTacticalGraphicsDef, 171
--mergeLeafUnitDef, 172
--mergeModelDefs, 172
--mergeTacticalGraphicsTypeMap, 172
--mergeUnitTypeMap, 172
--mimModule, 186, 1149
--modelPreloadFileName, 219
--msdlSIDCMappingFile, 182
--multicastAddresses, 177, 187, 192, 1151
-N, 151, 175, 185, 191-192, 1150, 1677
-n, 173, 182, 189, 1149, 1308, 1672
--netnFomRevision, 151, 186, 1149
--netnFomVersion, 151, 186, 1149
--noAppDataEntityDefs, 172
--noAppDataEntityTypeMap, 172
--noAppDataModelDefs, 172
--noAppDataSpotReportDefs, 172
--noAppDataTacticalGraphicsDefs, 172
--noAppDataTacticalGraphicsTypeMap, 172
--noAppDataUnitDefs, 173
--noAppDataUnitTypeMap, 173
--nonVrfObjectsUUIDScheme, 173, 182
--noRtiCompilerCheck, 186
--notifyLevel, 173, 182, 189, 1149, 1308, 1672
--numberOfremoteNetworkThreads, 176
--numberOfRemoteNetworkThreads, 177
--numCallbackThreads, 182
-O, 168, 323, 342
--outputInterval, 182, 1149
--outputLogFile, 182
--overwriteMappingsOnMerge, 172-173
-P, 152, 177, 186, 192, 1151, 1679

-p, 175, 185, 1150
--plugin, 173
--pseudoFullScreen, 1629
-q, 180
-r, 183
--recvBufferSize, 177, 182, 1150, 1679
--reportedHostAddressString (DIS), 176-177
--requestWhenInitialized, 176
--rprFomRevision, 151, 176, 186, 1150
--rprFomVersion, 151, 176, 186, 1150, 1678
--run, 155, 187-188
-s, 156, 176-177, 183, 189, 1150, 1675
-S, 174, 176, 186, 192, 1151, 1214
--scenarioFileName, 182, 251
--sendBufferSize, 177, 182, 1150
-sessionId, 133, 150, 152, 176-177, 183
--sessionId, 183
--setFomModuleList, 151, 186, 1151
--setMainThreadToHighPriority, 183
--settingsFile, 183
--sharedDataDir, 173, 1308
--sharedDataRoot, 173, 1308
--sharedDataStem, 173, 1308
--sharedSettingsDirectory, 173, 183, 1308
--showConsole, 173
--sim, 187
Simulation Object Editor
--, 1309
--appDataDir, 1306
--dataDir, 1306
--doNotCheckPluginVersions, 1306
--doNotLoadVrfPlugins, 1306
-G, 1306
-h, 1309
--help, 1309
--ignore_rest, 1309
--launcherLocation, 1307
--layoutSettingsFile, 1307
--lightMode, 1307
--loadPlugin, 1307
--locale, 1306
--settingsDirectory, 1308
--showConsole, 1308
--smsFile, 1308
--upgrade, 1308
--upgradeSystems, 1308
-v, 1309
--version, 1309
--visualTerrain, 1308
--simulationModelSet, 183
--siteId, 156, 176-177, 183, 189, 1150, 1675
--startMinimized, 184
--subnetMask, 177, 184
--suppressDataAvailableException, 184
--suppressHatIntersectOnAttach, 174, 1214
--suppressSelfReflect, 187, 1150
-T, 184
--tacticalGraphicsDefsDir, 174
--targetFrameRate, 184, 215
--testOutputFilename, 184
--timeManagement, 144, 186, 192
--tracyPort, 184
--unitDefsDir, 174
--useAbsoluteTimeStamps, 150, 152, 176-177, 184
--useAsyncIO, 177, 184, 193, 1150, 1679
--useFileCommChannel, 174
--useIpv6, 152, 177, 184, 1150
--userDataDir, 174, 184
--userScenarioDirectory, 174, 254
--userTerrainDirectory, 174
--useUserSettingsDirectory, 174, 185, 188, 1150, 1308
-v, 112, 175, 185, 188, 1151, 1309
--version, 112, 175, 185, 188, 1151, 1309
--waitQueue, 185
-x, 151-152, 175, 177, 185-186, 190, 193, 1150-1151, 1677, 1679
DIS, 193
HLA, 190
-z, 165
-Z, 168
command-line option, ignore rest, 112, 175, 185, 1151, 1309
comma-separated values file, See CSV file
comment, MTL parameter, 1664
comm-model-name parameter, 1574
commModelParams.mtl file, 1574
communication
component, 366
file, 174
jamming, 539
line-of-sight, 1073, 1076
communication model, 1574
custom, 366
simulation object, 365
company (unit)
pre-configured, 440
simulation object icon, 1080
comparison operator, 942
compass
color, 1181
configuring, 1180
degrees, 1179
keyword, 1613
location, 1181
measurement unit, 1634

model definition, 1180
NATO-MIL, 1179
observer, 1178
component, 1472
actuator, 367
architecture, 366
attaching to remote entity, 230, 1592
communication among, 366
controller, 366-367
descriptor, elements of, 1478
emitter, 878
entity, 1477
sensor, 366
simulation object, 1477
systems, 1482
tuning performance, 206
component attachment table, 1592
Component Attachment Table parameter, 232
Component Manager, 1685
Component-Attachment parameter, 353
Component-Attachment-Table parameter, 353
componentAttachmentTable.mtl, 1592
component-descriptor-type, 1478
component-manager parameter, 1685, 1687
component-type, 1478
composing
terrain, 1242
compute
aerodrome, 1293
computed route, 1019
Concealed set data request, 873
concurrent task, 554, 561, 765, 771
condition
considerations for using, 939
Detect Entity, 939
Do Until Interrupt, 938
Engineering Object Breached, 939
Entity Altitude, 939
Entity Created, 939
Entity Destroyed, 939
Entity Di-Guy Animation Check, 939
Entity Di-Guy Appearance Check, 940
Entity Embarked, 940
Entity Has Target, 940
Entity In Area, 940, 1016
Entity In Range of Target, 940
Entity Left of Line, 940
Entity Under Fire, 941
If, 935, 950
Lifeform Surrendered, 941
Missile Approach Warning, 941
name, 963-964
pattern, 963-964
editing CSV file, 964
Random, 941
Receive Text Message, 942
Resource, 942
Scenario Event, 942
Simulation Date/Time, 942
Simulation Object Independently Tasked, 942
Simulation Time, 942
Tactical Graphic Active, 942, 1035, 1038
Wait Until, 938, 951
While, 937, 950
conditional expression, 956
conditional statement, 933
building, 956
filtering simulation object list, 565
trigger, 935
condition-entity-types.csvfile, 964
cone
gimbaled sensor, 1229
sensor color, 899
configuration
DI-Guy, character and appearance, 1464
display, 165
display engine, 1598
loading, 1602
saving, 1602
embarkation view, 1443
entity display, 166
environment, 167
file, See configuration file
joystick, favorite, 1588
MAK radio, 1154
MAK RTI file, 107
plug-in
adding, 160
deleting, 161
simulation model set, 235
world cell size, 167
configuration file
character_animations_table.mtl, 1464
commModelParams.mtl, 1574
condition-entity-types.csv, 964
for object parameter database, 1309
navigation area, 1287
sysdef, 1482
vrfSim.mtl, 1668
vrfSimSettings.xml, 212
configuring
joystick, 1579
keyboard entity control, 1579
menu, 1640
toolbar, 1645

confirmation prompt, enabling and disabling
plan, 561
task, 561
connection
automatic, 153
configuring in system definition file, 1483
information, 153
sensor, 1504
simulation object group, 1363
to network, automatic, 164
console
configuring for back-end, 1670
messages, 189, 972
notification level, 890
object, 483
constrained time, 142
constraints, view, 1200, 1214
Construct Abatis task, 720
Construct Anti-Tank Ditch task, 721
Construct Barbed Wire task, 721
Construct Barricade task, 722
Construct Berm task, 722
Construct Booby Trap task, 722
Construct Bridge task, 723
Construct Dry Gap task, 723
Construct Fortified Area task, 724
Construct Fortified Line task, 724
Construct Minefield task, 725
Construct Strong Point task, 725
constructing terrain, See also composing, terrain
constructive model, See aggregate-level
contamination, 532
agent, 532
biological, 716
chemical, 717
nuclear, 718
type, specifying, 918
Contamination set data request, 918
context-sensitive menu, 1630
configuring, 1640, 1643
contrail, 1051
contrast, 1117
control object, 994, 1687
concepts, 994
copying, 395-396
count of, 487
creating, 372
editing, 1029
list of for scenario, 357
name, 994
parameter-type, 1501
pasting, 395
pedestrian path, 435, 994
selecting, 455
spot report, 295
control panel
entity, 567
gimbaled sensor, 1230
control, remote, 87
controller, 366
component, 366-367
detonation, 1536
dynamic-terrain-munition-damage, 1560
game, 1578
indirect fire, 1534
indirect fire weapon, 1534
joystick, 1579
lasing, 1533
launcher, 1533
making decisions and performing tasks, 366
missile, 1533-1534
target selection, 1532-1533
conventional mines, 1021
convoy, 586-587
behavior, 572
disallows aggregation, 1407
Convoy Along task, 572, 586
Convoy To task, 572, 587
coordinate
point, 1262
coordinate system, 84
3D, 1182
3D Cartesian, 802
Cartesian, 802-803
changing, 1634
GARS, 1635
geocentric, 1246
Lambert Conformal Conic, 1182
LCC, 1182
supported, 1245
UTM, 1182
coordinates
going to, 1207
observer view, 1177
copying
formation, 1414
objects, 395-396
performance and, 396
plan statements, 981
scripted task, 779
spawn pattern, 426
copying a key mapping, 1654
corridor, 1016
wind, 339
count
object, 487

simulation object, 487
countermeasure
ammo select table, 311
freeze levels, 881
launching, 311, 640
set data request, 874
Countermeasures Auto Launch set data request, 311, 874
country
code, 1669
marking, 1357
simulation object, 1342
countryCodesMappingFile parameter, 1669, 1672
cover
rain accumulation, 329
snow on ground, 329
CPU time, giving up, 215
Crank Left task, 587
Crank Right task, 587
crash, application, 99, 1660
crawling, human posture, 892
Create global command, 973
Create More Entities task, 652
Create Object dialog box, 384
configuring, 384
Create Object Palette, 373-374
image, 376, 1362
Create Sensor View command, 978
Create_Global_Dynamic_Terrain parameter, 355
Create_Global_Environment parameter, 355
Created Object variable, 943
create-script-id parameter, 768
creating
booby trap, 722
combat engineering object, 535
demonstration recordings, 975
fortified area, 724
fortified line, 724
key map, 1655
new entity in Simulation Object Editor, 1360
prop, 402
window layout, 1624
crepuscular ray, 1095
crew, setting for air base, 923
criss-cross applesauce, 635
crosswalk, 436
crouching, human posture, 892
crowd
adding members, 652
creating, 432, 434
disallows aggregation, 1407
dispersing, 655
gathering at location, 652
gathering at waypoints, 653
performance, 881
protesting, 661-662
rioting, 662
tasks, 649
unit, 432
wander, 665
Crowd Along Line task, 652
Crowd Around task, 652
Crowd Around Waypoints task, 653
Crowd in Front of Entity task, 654
cruise missile
configuring, 1532
firing, 633
CSV file
animations, 1399, 1401
ballistic missile, 319
configuring for ballistic missile or animated movement, 1403
importing objects from, 243
cube map, recalculating, 1096
culling
face front, 1112
cultural feature, 400, 1377
aligning to gravity, 1382
creating, 372-374, 377, 382
embedding in terrain, 401, 1381
ground clamping, 1382
heading, specifying, 389
information, 462
pitch, specifying, 389
cultural-feature-param parameter, 1476, 1688
cultural-feature-state-repository, 1476
Current Speed set data request, 874
cursor, 381
curvature of the earth, display mode, 1076
custom
overlay name, 1341
plan, 416, 421, 425
customization, 157

D

daemon
running vrfSim as, 1669
starting back-end as, 180
daemon parameter, 1669
damage
actuator, 1551
calculating, 1546
displaying, 1353
dynamic terrain, 1560

file, 1502
model, 1390
specifying, 1392
probability table, 1531
damage file, 1527, 1546
damage-by-power, 1551
Damage Percentage set data request, 875
damage system, 1527, 1546
migrating to new release, 1322
damage-by-munition-power block, 1551
damage-by-power, 1531, 1546, 1554
adding new, 1553
configuring, 1546
damage file, 1551
terrain, 1560
damage-by-power-entry, 1551
damage-model block, 1551
dark mode, enabling and disabling, 1637
dashed line, 1033
data
catastrophic kill, 1558
configuring frequency of sends, 1670
directory, 103, 164, 178, 187, 1306
sim. engine, 180
specifying location of, 164, 180, 187, 1306
effect, power type, 1558
elevation, 1243
firepower kill, 1558
input, 1477
install, 96
options, 101
mobility kill, 1558
navigation, 1284
path, 1272
port, 366
shared
directory, 173
root, 173
shared stem, 173
synchronizing, 126
user directory, 174, 184
data type state property, 1492
database, 1245, See also terrain database
large, 1246
loading, 1246
object parameter, 1468
paging, 1247
terrain, 84
loading, 144
terrains in MAK ONE data, 1238
database (SQL), logging simulation events, 88
date, 330
format, 331
setting, 331
simulation, 236
datumShiftX parameter, 1669
datumShiftY parameter, 1669
datumShiftZ parameter, 1669
daylight, controlling light point, 1092
Deactivate Jammer task, 737
deactivation
indirect fire, 1035
jammer, 737
spawn points, 418
dead-reckoning, 213, 1673
changing for fast forward, 261
debug
MAK Remote Debugger, 223
Navigation Lab, 1670
terrain, 223
debug overlay, 221
debugging
callstack.log file, 98
dmp file, 98
file location, 99
decal effects, 1051
enabling and disabling, 1051
models, 1052
decay rate for spot reports, 294
decibel, configuration, 1144
decision making, 366
decreasing speed of navigation, 1200
deep-snow-depth parameter, 507
default
formation, 1413
joystick configuration, 1588
name, 382
overlay, specifying, 1341
parameter, 358
plan, 416, 421, 425
Set commands, 861
settings, restore, 123
simulation model set, 164
configuration, 235-236
spawn pattern, 422
value, parameter, 358
variable binding value, 1487
view, 1205
default_guiConfiguration.gui, 1493, 1640
default_guiSimpleScenarioPlayer.gui, 1640
default_ModelPreload.mpsx file, 218
DefaultEllipsoidArc model definition, 1136
defaultHeartbeatThreshold parameter, 1678
defaultObjectTimeout parameter, 1678
DefaultObserverView parameter, 355

defaultParameterDatabasePath parameter, 356, 1669
defaultSpawnTemplates.mtl, 422
default-task-rules.tsk, 580, 765
defaultTerrainDatabasePath parameter, 356, 1670
defense parameter, 1436
defensive
  measure, 640
  movement, 734-735
definition
  entity, directory, 166
delay parameter, 1561
Delete Object global command, 973
Delete Self command, 422
Delete Self global command, 973
deleting
  articulated parts, 1389
  assembly, 1428
Deliberate-Attack posture, 530
Deliberate-Defense posture, 530
DEM, 1243
demonstration
  creating, 975
  mode, 1629
Deploy Port Refueling Drogue task, 701
Deploy Refueling Boom task, 700
Deploy Sonobuoy task, 705
Deploy Sonobuoys Along Route task, 705
Deploy Starboard Refueling Drogue task, 701
deploying
  embedded entity, 703
    with task, 704
  sonobuoy, 705
depth
  bits, 164
  buffer, 207
  focal, 1233
  moving to, 606
  rain, 329
  snow, 329
  sonar, 900
depth charge, 625-626, 631
depth of field, 1233
  configuring, 1235
  enabling and disabling, 1234
derived SMS, 1310
  hiding included objects, 1314
  promote object from included, 1314
  promote system from included, 1316
descent rate, 591
description
  script, 754
  simulation object, 364
  trigger, plan, 952
deselecting, See unselecting
designator, 364
  lasing, 304
Designer, 854
destAddrString parameter, 1678
destination address, 152, 176, 180, 192, 1149, 1678
Destroy Bridge task, 726
Destroy Ditch task, 726
Destroy Explosive task, 727
destroy federation execution, 1677
Destroy Fortification task, 727
Destroy Obstacle task, 727
Destroyed set data request, 875
destroyFedExec parameter, 1677
destroy-script-id parameter, 768
destructible terrain, 1238
detaching observer from object, 1213
detachment (unit)
  simulation object icon, 1080
Detect Entity condition, 939
Detected CID level, 302
detecting target, 302, 1528
detection table, 303, 1516
detection types, sensor, 1512
detection-types parameter, 1528
determinant
  coefficients, 1553
  direct, 1549
  equivalent-tnt-mass, 1549
  power, 1547
  range-coefficients, 1549
  range-list, 1549
determinant block, 1551
detonation
  car bomb, 875
  controller, 1536
  effects, 1057
  event, sound effect, 1142, 1145
  IED, 875
  interaction, 1534
  line, 1058
  power, configuring, 1547
  proximity, 1536
  result, 1562
Detonation Fuse Type set data request, 875
detonation-munition parameter, 1561
detonation-on-destroy parameter, 1561
detonationParams.mtl, 1535, 1546, 1551, 1553, 1562
detonation-power-entry parameter, 1546
developer debug overlay, 221

developer documentation, 97
Developer's Guide, installing, 97
device address, 150-151, 177, 180
deviceAddress parameter, 1678
diagnostic
  level, 173, 182, 189, 1149, 1308, 1672
dialog box
  configuring, 1641
    pages, 1640
  Information
    configuring, 1642
    displaying state properties, 1493
  locking, 1647
diffuse color, 1089
digging anti-tank ditch, 721
DI-Guy
  action table, 1465
  adding new characters, 1464
  animation
    adding, 1465
    choosing, 1466
    testing for, 939
    urgency, 908
  appearance, testing for, 940
  character and appearance, configuring, 1464
  character data file, 180
  configuring, 1670
    animations and appearances, 180
    character and appearance, 1383, 1386
  displaying in vehicle, 1085
  hand item, stowing, 902
  instancing, 220
  integration with VR-Forces, 654
  license, 89
  motion file, 1465
  performance, 220
DI-Guy Animation task, 654
DI-Guy Characteristics set data request, 876
DI-Guy Gesture task, 654
DI-Guy Hand Item set data request, 876
diGuyAnimationsFile parameter, 1670
dip sonar, 706
direct determinant, 1549
direct fire weapon
  configuring, 1527
  firing, 632, 636
direction
  line, changing, 1033
  Lua, 801
  tidal current, 341, 345
  wind, 324
directory
  data, 103
  installed, 102
  sim. engine, 180
  Windows default install, 97
DIS, 129
  command-line options, 192
  configuring, 1678-1679
  Enumeration Document, 119
  exercise ID, 152, 177, 186, 193, 1151, 1679
  heartbeat threshold, 1678
  port, 152, 177, 186, 192, 1151, 1679
  port number, 192
  self reflection, 187, 1150
  specifying broadcast modes, 192
  starting in, 164
  startup tutorial, 134
  timeout, 186
  version, 152, 177, 186, 193, 1151
DIS Enumerations Document, 1469, 1562
disable_if_checked, 1489
disable_if_multiple_selection, 1489
disabled-by-suppression parameter, 522
disableParallelPublisherTick parameter, 1670
disableParallelTick parameter, 1670

-

--disableRemoteDiscovery command-line option, 177
Disembark All task, 681
Disembark command, 410
Disembark task, 681
Disembarked set data request, 410, 877
disembarked status, testing for, 940
disembarking, 410
  all entities on parent, 410
  entity, 406, 877
  instantly, 410
Dismount Headquarters task, 669
Dismount Platoon task, 670
Dismount Squad task, 670
Dismount task, 691
Dismount Team task, 671
Dismounted Headquarters Section set data request, 909
Dismounted set data request, 909
dispersal-angle parameter, 629
Disperse Crowd task, 655
display
  dragging, 1189
  field of view, 1609
  loading configuration, 165
  navigating, 1177

perspective, 1609
remote, 217
time, format, 332
unit of measurement, 1634
display engine, 1598
configuration, 1598
loading, 1602
saving, 1602
window, adding, 1598
Display Engine Configuration Editor, 1598
Display Terrain parameter, 231
displaying
bounding volume, 1353
entity information, 469
environment setting, 1353
feature data, 1256
full screen, 1629
grid line, 1252
height-above-terrain lines, 480
hidden panels, 1624
intervisibility
lines, 1129
object, 1126
label, 473
entity, 470
laser designator, 1070
objects, 998
other MAK 3D GUI viewers, 1172
panel, 1624
performance statistics, 202
persistent tracks, 1053
scenario information, 252
segment outlines, 1139
stealth models, 1173
tactical graphics, 999, 1035, 1038
toolbars, 1624
track histories, 1054
trailing effects, 1051
transient intervisibility, 1125
vertex labels, 1000
view, 1204
visual effect, 1354
disPort parameter, 1679
distance
specifying unit, 1634
view with bullseye, 1026
view with range line, 1130
Distance To Attached policy, 1607
Distributed Interactive Simulation, 93, See also DIS
ditch
anti-tank, digging, 721
constructing, 723
destroying, 726
dynamic terrain, 1023
improving, 728
DLL, 157
plug-in
adding, 159
removing, 160
specifying for plug-in, 159
dmp file
debugging, 98
location, 99
Do Until Interrupt statement, 933, 938
docking
menu, 1628
panel, 1622
toolbar, 1623
vertically, 1623
documentation
help, 66
installation, Windows, 97
documentation, location of, 104
domain
power, 1547
sensor, 1513
doNotUseConsole parameter, 1670
door
building
closing, 651
opening, 660
car
closing, 699
opening, 698
cargo
closing, 700
opening, 699
Doritos ®, 486
dotted line, 1033
Douglas sea state, 341, 344
drag chute, setting for aggregate air unit, 921
drag-and-drop
canceling move, 393
plan statements, 981
dragging
canceling move, 393
object to new location, 392
objects, 887
simulation object, 392
drainInput
configuring, 1672
for DIS, 1679
for HLA, 1677
dr-allow-gui-overrides parameter, 213
draw call, coloring, 215

drawing
  arc, 1014
  area, 1012
  box, 1016
  circle, 1013
  ellipse, 1013
  spider web, 1015
driver, vehicle
  embarking as, 683
  slot, 683
driving
  left or right side or road, 504
  on roads, 503
drogue refueling, 701
drone, munition
  armed, 870
  detonation fuse type, 875
  loiter attack, 643
  target, 905
Drop Fuel Tanks task, 738
Drop Naval Depth Charge at Location task, 626
Drop Naval Depth Charge task, 625
Drop Naval Mine task, 626
Drop Naval Mines Along Route task, 628
Drop Off Passengers at Location task, 669
drop slots
  number available, 585, 683
  specified for entity, 684
dropping
  bomb, 646-647
  mine, 626, 628
dropping from exercise, 118
dry gap
  building, 723
  construct, 723
  destroying, 726
  improving, 728
DtCgfDispatcher class, 358
DtDetonationManager class, 1562
DTED
  elevation, 1243
DtSimComponentManager class, 1473
DtSimObjectNetInterface class, 1473
DtSimObjectStateRepository class, 1473
DtSimTaskManager class, 1473
DtVrfObject class, 1469, 1473
DtVrfObjectBaseStateRepository class, 1469
DtVrfObjectParameters class, 1469
DtVrfObjectPlanManager class, 1473
DtVrfRadio class, 1574
DtVrfRemoteController class, 358
dump file, See dmp file
duplicating, See copying
dust
  blowing in the wind, 337
  cloud, 1051
dynamic avoidance, 1276
dynamic lighting, 1090
dynamic linked library, 157
Dynamic Near Clip Attached Policy attribute, 1607
dynamic ocean, 86, 341
  enabling and disabling, 342
  rotor wash, 1109
  wake and spray effects, 1110
dynamic terrain, 165
  berm, 1023
  damage, 1560
    high fidelity, 1561
  ditch, 1023
  global dynamic terrain processor, 232
  navigation data regeneration, 1284
  tank scrape, 1023
Dynamic Water Visibility Enabled attribute, 1611
Dynamic Water Visibility Maximum attribute, 1611
dynamic-terrain-munition-damage controller, 1560
dynamic-terrain-type parameter, 1561

E

earth file, 90, 1244
  hiding layers, 1255
  limit splatting textures, 168
  limit terrain textures, 169
  limit vegetation textures, 169
echelon ID, 362-364
  assignment of, 365
  unit member, 516
echelon level, ghosted icons, 446
echelon-level parameter, 1501
ECW, 1243
editing
  channel attributes, 1604
  environment condition, 335
  frustum, 1609
  intervisibility object, 1126
  key maps, 1650-1651
  object
    in OOB, 494
    scene, 1030
  observer mode, 1167
  parameter, 1380
  window attributes, 1603

effect
data, power type, 1558
fire and detonation, 1057
lighting, 86
object, creating, 374
ocean, 341
sensor, new, 1119
trailing, 1051
egress point, 1442, 1569
configuring, 1451
egress technique, 1450
egress-points parameter, 1569
electromagnetic emissions, 878, 1072
segment angle, 1138
sensing, 541
electronic
emission, sensing, 539
warfare, 306
aggregate level, 539-541
entity level, 866, 885, 895
warfare parameter, 1437
element
informationitem, 1643
informationpagecollection, 1642
mainmenu, 1640
mainmenuconfiguration, 1640
menuitem, 1644
pagecollection, 1641
pagecollections, 1641
rightclickmenu, 1643
toolbar, 1646
toolbarrow, 1646
element definition
changing type, 1356
common buttons, 1630
exporting, 1359
importing, 1359
pruning, 1359
spot report, regenerating, 1349
element_type, 1489
elevation
data, 1243
intervisibility point, configuring, 1127
snow cover, 329
ellipse
drawing, 1013
extruded, 1016
rotating, 1032
undoing and redoing rotation, 1032
ellipsoid, reference, 1669, 1675
Embark On command, 407
Embark task, 407, 681
embarkation
automatic, 409
condition, 940
configuring, 1442
slot exclusions, 1449
view, 1443
editing ingress and egress points, 1569
egress point, 1451
embedded entity alternative, 411
immediate, 407
ingress point, 1451
instant, 409
occupancy-director-controller, 1569
posture, 1447
slot, 1569
configuring, 1444, 1451
ingress and egress technique, 1450
Embarkation View, 407, 452
embarking objects in, 408
embarkation-slot-entry parameter, 1569
embarkation-slots parameter, 1569
Embarked set data request, 407, 877
embarking
entities, 386, 406
humans in cars, 683
in Embarkation View, 408
embedded entity, 411, 703
assigning tasks to, 413
configuring, 411
deploying, 703
deploying with task, 704
recovering, 703
restoring parent, 414
Simulation Object Editor, 411
system, 411
embedded window, 1598
EM-Emission update message, 1137
Emergency Beacon set data request, 878
emission sensor, 541
emissions-sensor system, 541
emitter, 894
beam, 310, 878
configuring, 1572
ID, 878
jammer, 885
information, 463
power, 1137
setting, 878
systems, 1072
Emitter set data request, 737, 878
emitter volume, 1136
color, 1136
outlines, 1139

pulse rate, 1136
radius, 1137
segment angle, 1138
segment size, 1137
empty units, 1407
enable_if_checked, 1489
enableLogFileTimestamps parameter, 1670
enableNavigationLabDebugging parameter, 1280, 1670
enableVSync command-line option, 216
enabling
marine effects, 342
sound effects, 1142-1143
time management, 143
in sim. engine, 144, 186, 192
ending
scenario, 263
scenario event, 285
endTask() function, 796
Engage From Soldier Firing Positions task, 677
Engage From Vehicle Firing Positions task, 677
engagement
information, 465
rules, 897
tasks, 623
zone, 305, 870, 879
joint, 1016
missile, 1016
Engagement Results set data request, 919
Engagement Zone
set data request, 879
engineering object
information, 536
percent complete, specifying, 923
Engineering Object Breached condition, 939
entity, See also simulation object
3D model scale, 1082
add to Props Palette, 402
adjusting clipping plane, 386
aggregate-level
creating, 1418
parameters, 1430
aircraft
attacking, 625
attacking with gun, 625
clutter, 428
altitude
displaying, 480
specifying, 387
testing, 939
animation, 585
attaching observer to, 1210, 1214
attaching tactical graphic to, 871
background, 428
body frame coordinate system, 1182
bounding volume, 481-482
centering observer on, 1197
clamping to ground, 1248
clutter
creating, 428
promoting, 893
collision avoidance, 873
component, 1477
attaching to remote, 1592
configuring
emitter, 1572
obstruction avoidance, 1568
console message, 483
controlling by keyboard, 1579
controlling with joystick, 1578
copying, 395-396
count, 487
creating, 372-374, 377, 381-382
from existing, 1361
in plan, 973
in Simulation Object Editor, 1360
culling, 167
cultural feature
embedding in terrain, 401
definition directory, 166
deleting, 399
from plan, 973
from simulation model set, 1362
disembarking, 406, 877
all, 410
display settings, 166
dragging, 392
editing, 393, 1305
element definition
prevent loading, 172
embarkation
configuring, 1442
testing, 940
embarking, 386, 406
embedded, 411
enumeration
changing, 1338
condition, 963
in plan, 964
exclude textures, 169, 1307
extents, zooming on scenario load, 1195
file, importing, 1326
filtering
attachment list, 1213
fixed-wing, 510

following
entity, 593
route, 599
watercraft, 595
following route with actions, 600
generating navigation data, 1285
ghosted, 446
ground
clamping, 1249
road driving, 503
heading, specifying, 389
icon
changing size, 1080
image, adding, 1351
in unit, 440
indirect rendering
disable, 165
inset view, 1224
instrument panel, 1228
intervisibility, 1123
label, 473
2D, 470
3D, 469
color, 469
enabling and disabling, 471-472
pinning to display, 472
leaf file
merge, 171
level body frame coordinate system, 1182
limit textures, 169, 1307
list, filtering, 456
mapping sound effects to, 1145
messages
notify level, 483
viewing, 484
model, 1309
adding visual, 1351
editing, 1337
editing visual, 1346
movement, scripted, 1399
moving to
altitude, 605
location, 604
object or entity, 603
name, 469-470
changing, 393
pasting, 397
notification level, object console, 483
object geometry, 1380
organization, 364
pasting, 396-397
patrolling between, 609
patrolling route, 609
pitch, specifying, 389
placing, 382
new, 386
on entity, 386
plan, 932
platform, updating, 1326
primary, 1216, 1219
properties, specifying at creation, 384
random creation of, 400
range ring, 1068
reinitializing, 399
repositioning, 391
resource
at creation, 391
monitoring, 481
saving console messages, 485
scenario, importing to, 238
secondary, 1216, 1219
selecting, 453
shadows, 1100
ship, clutter, 429
sound effects, 1142
spawning, 416
maximum simultaneous, 423
specifying
target, 905
surface
buoyancy, 1111
symbol color, 1333
system, 1390
disabling, 902
enabling, 902
status, 463
tag, editing, 1337
task
keyboard shortcut, 571
visualizing, 1060, 1062
trailing effects, 1051
type mapping file, 167
merge, 170
prevent loading, 172
type, changing, 1338
type, enumeration, 470
unit, 438
velocity, publishing, 1673
visual markings, 888, 1357
VR-Forces, 120
wake
configuring, 1395
enabling and disabling, 1110
zooming to extents, 1194
Entity Altitude condition, 939
Entity Control Panel, 567

Entity Created condition, 939
Entity Destroyed condition, 939
Entity DI-Guy Animation Check condition, 939
Entity DI-Guy Appearance Check condition, 940
Entity Editor, See Simulation Object Editor
Entity Embarked condition, 940
Entity Has Target condition, 940
Entity In Area condition, 940, 1016
Entity In Range of Target condition, 940
Entity Left of Line condition, 940
Entity Under Fire condition, 941
entityDestroyed() function, 789
entity-level modeling, 368
EntityLevel.sms, 368
entity-range parameter, 1530
entity-type parameter, 1530
entrance, embarkation, 1451
entry point, scripted task, 787
entVrfDataTimeThreshold parameter, 1670
enumeration, 1469
  entity, changing, 1338
  in plan condition statement, 963-964
  object type, 1688
  published and matching, 1470
environment
  condition, See environment condition
  disabling, 1100
  global, 232
  loading configuration, 167
  weather zone, 338
environment condition, 324, 333
  adding, 335
  deleting, 335
  editing, 335
  marine settings, 343
  rain puddles, 329
  snow on ground, 329
environment effects, previewing on entity, 1353
environment variable
  MAK_INSTALL_DIR, 97
  MAK_LOG_DIR, 99
  MAKLMGRD_LICENSE_FILE, 105
  OpenSceneGraph, anaglyphic stereo, 1617
  RTI_CONFIG, 107
  VRV_ALPHA_TO_COVERAGE_RENDERBIN, 208
ephemeris, 232
  model, 331
equipment
  pacing and tracking, 919
  unit, 1420, 1422
Equipment Pacing/Tracking set data request, 919
equipment parameter, 1438
equivalent-tnt-mass determinant, 1549
error message, disable, 165
Escort Air Unit task, 733
escort, waiting for, 736
Establish Refueling Station task, 587
event
  actuator, 367
  indirect fire, 318
  linked, 280
  list order, 287
  scenario, 276
  show or hide, 287
EW-Comms-Degradation-Percentage, 539
EW-Comms-Dependence, 539
EW-Defense-Strength, 539
exaggerated reality, 1171
exaggerating
  3D models, 1082
example scenarios, 254
exclusion slot, 1449
execName parameter, 1677
executable
  installation location, 102
    Windows, 97
execute ground attack, 630
Execute Surface Attack task, 630
execution name, HLA, 190
exercise
  clock, 120
  site ID, 156, 176-177, 183, 189, 1150, 1675
  starting, 260
exercise ID, 1679
  DIS, 152, 177, 186, 193, 1151, 1679
exerciseId parameter, 1679
exercise-start-time parameter, 354
exit, embarkation, 1451
exiting
  application, 146
  full screen window, 1629
  plan, 982
  prompt, 146
  scenario, 263
  simulation engine, 146
expand unit, 444
expanded label, 469
  pinning to display, 472
expectedLocalEntityCount parameter, 1677
Explode Charge at Depth task, 631
explosion, 1058
explosive device
  arming, 870
  system, 1536
explosive power, 1535
  domain, 1547

exporting
element definitions, 1359
model definitions, 1359
MSDL, 241
object information, 1327
object type mappings, 1359
Order of Battle, 495
overlays to shapefiles, 1008
scripted task, 777
settings, 123
tags, 1330
visual models, 1359
Extend Towed Array Decoy task, 702
extended label, 477, 995
adding, 478
deleting, 479
index, changing, 478
extended name, 754
extensions, 157
extent, zooming to, 1194
external communication model, 1574
external renderer
indirect, disable, 165
external script, 782
Extract Using Ladder task, 690
Extract Using Rope task, 687
extracting
prop from terrain, 1244
extras file, 353
extruded
area, 940
tactical graphic, 1016
extruded geometry, 1244

F

face front culling, 1112
Facilities set data request, 919
factory settings
directory, default, 167
restoring, 123
window layout, 1628
fan, intervisibility, 1123
far clipping plane, 1605
FASCAM task, 738
Fast Rope Insertion task, 686
fault tolerant heap, disabling, 1661
favorite, 377
adding or removing from list, 402
joystick configuration, 1588
observer view, 1205
favorites.mst, 402
FDD file, 107
feature
avoiding, 1568
configuring avoidance of, 1568
coordinate, 1262
data, displaying, 1256
geometry, 1244
indexing, model layer, 1662
soil type, 1262
vector list, 1258
feature layer, 1244
featureconfig.txt, 1568
FeatureSet class, 784
FED file, 107
name, 151, 175, 185, 191-192, 1150, 1677
specifying, 192
federate
name, 175, 185, 1150, 1677
time managed, 142
type, 175, 185, 1150
federateName parameter, 1677
federateType parameter, 1677
federation execution
destroying, 1677
name, 1677
specifying, 190
federation time, 143
fedFileName parameter, 1677
FFE, See Fire for Effect task
field of view, 1137, 1191, 1609-1610
changing, 1609
file
back-end configuration, 1668
batch, 269-270
communication, 174
damage, 1546
default_ModelPreload.mpsx, 218
FDD, 107
FED, 107, 192
name, 151, 175, 185, 191-192, 1150, 1677
formation, 1566
hostility relationships, 358
installed, 102
log, 162, 1670
MTL, 1664
object map, 357
OPE, 1520
physicalWorldParams.mtl, 1513
plan, 356
platform, 1481
RID, 107, 190
rid.mtl, 190
scenario, 225, 350-351, 355
extras, 422

objects, 350, 357
scnx, 225, 350
simulation model set, 1309
system definition, 1520
transporter (DIS), 175, 177
vrfGui.log, 162
vrfSim.log, 162
vrfSim.mtl, 1668
vrfSimSettings.xml, 212
Fill Style set data request, 1040
fill style, tactical graphic, 1033
filtering
earth file layers, 1255
function list, 1655
object attachment list, 1213
object creation palette, 377
object lists, 456
scripted task list, 776
simulation object list, 565
Find Cover task, 655
Find Firing Position task, 656
finite state machine, 786
fire
effects, 1057
event
mapping sound effects to, 1145
sound effects, 1142
indirect, 314
line, 1058
suppressive, 637, 645
at location, 520
at point, line, or area, 520, 637, 645
configuring, 521
target, 632, 635-636
fire and maneuver task, 657
Fire At task, 632
Fire Ballistic Gun task, 632
Fire Cruise Missile task, 633
Fire for Effect task, 634
fire interaction, 1532-1533
Fire Javelin at Target task, 635
Fire Laser Guided Missile task, 636
Fire Missile Launcher task, 636
fire team
dismount, 671, 909
dismount task, 691
mount, 672, 910
Fire Team Bound to Location task, 657
Fire Team Occupy Fire Position task, 658
Fire Team Suppress Target task, 520, 637
Fire Team Travel to Waypoint task, 658
fire-when-fired-upon, 897
Fixed Near Clip attribute, 1607
Fixed Near Clip When Attached attribute, 1607
fixed trajectory missile, 319
fixed weapon system, configuring, 1536
Fixed Wing Land At Aerodrome task, 512
Fixed Wing Land On Runway task, 512
fixed wing takeoff from route task, 619
Fixed Wing Takeoff On Runway task, 511
fixed-frame, 143
best-effort, 122, 354
block on asynchronous operations, 1669
keyword, 354
fixed-frame-run-to-complete, 122, 143, 354
fixed-wing entity
altitude, 589, 591
attacking, 625
attacking with gun, 625
circling a point, 607
countermeasures, 640
crank maneuver, 587
creating, 510
dropping bombs, 646-647
follow entity behavior, 594
heading, 591-592
holding pattern, 610
hover, 594
land at airport, 596
landing, 512, 596-598
loiter pattern, 589-590, 592
measurement unit, 1634
refuel, 610
refuel self, 692
refueling station, 587
search and rescue
fly creeping line pattern, 613
fly expanding square pattern, 615
fly parallel pattern, 616
sector search, 618
setting
altitude, 510
IFF transponder, 883
speed, 464, 473
strafing target, 628
takeoff and landing, 511
taking off, 511, 619-621
fixed-wing-entity-param parameter, 1476, 1688
fixed-wing-entity-state-repository, 1476
flag
marking, 1357
flaming effects, 1057
flare, 640
launching, 311
flashlight mode, 1097
flat earth coordinate system, 1182

Flee From Entities task, 659
FLEXlm, 104
flight command, 589, 591-592
flight station, 1598
FLIR, 1116
floating panel, 1622
flock, See herd
FLT file, 1243
Fly Altitude task, 589
Fly Cloverleaf Pattern task, 589
Fly Creeping Line Pattern
  Search and Rescue task, 613
Fly Expanding Square Pattern
  Search and Rescue task, 615
Fly Figure 8 task, 590
Fly Heading and Altitude task, 568, 591
Fly Heading task, 592
Fly Loiter Pattern task, 592
Fly Parallel Pattern
  Search and Rescue task, 616
flying
  altitude to, 589, 591
  loiter pattern
    circle, 592
    cloverleaf, 592
    figure eight, 592
    hold pattern, 592
    racetrack, 592
  to heading, 591-592
focal depth, 1233
focal range, 1233
focus, scene, 1233
fog, 333
  denseness of, 327
  height and color, 328
  weather zone, 337
fog of war, 167, 290
  same side, 293
folder
  adding scripted task to, 777
  removing scripted task from, 777
  scripted task, 777
    adding, 777
    deleting, 777
    renaming, 777
Follow Along Offset Route task, 673
Follow Entity task, 593
  functionality, 505
  in plan, 945
follow entity, fixed-wing entity behavior, 594
Follow observer mode, 1182, 1214-1215
Follow Route task, 599
following
  paths, 1274
  route, 599-600
  simulation objects, 593
  watercraft, 595
FOM, 106
  module, 151, 175, 185-186, 1151
FOM Mapper
  configuring, 1677
  initialization data, 175, 185, 1150, 1677
  library, 175, 185, 191-192, 1151, 1677
  specifying at startup, 191
fomMapperInitData parameter, 1677
fomMapperLib parameter, 1677
food, 537
  pacing and tracking, 921
  specifying, aggregate-level, 920
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking set data request, 921
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant set data request, 920
footprint, 1051
  affect of posture, 530
  sensor, 533
  unit, 529
force, 364, 470
  adding, 1331
  behavior set, assigning to, 580
  color
    changing, 1333
    tactical graphics, 1001
    text, 474
  editing, 1333
  hostility
    changing, 1333
    changing in plan, 302
    editing, 300
    matrix, 300
  name, changing, 1333
  removing, 1333
  setting at runtime, 879
Force set data request, 879
  tactical graphic, 1040
forceHostilty.mtl, 301
forceParameterDbReload parameter, 1670
format
  time display, 332
formation, 517
  adding, 1412
  closing, 517
  collapsing in Formation Editor, 1413
  configuring, 1566
  copying, 1414

default, 1413
editing, 1411
expanding in Formation Editor, 1413
file, 1566
layout
automatic, 1415
manual, 1415
moving to, 603
removing, 1413
renaming, 1413
setting
in plan, 945
snap to grid, 1416
specifying, 879
supported, 879
tactical, 1411
transition to, 621
user-defined, 1567
formation bounding volume, 1414
Formation Editor, 1413, 1415
Formation set data request, 879
in plan, 945
fortification
breaching, 719
destroying, 727
improving, 729
fortified area, constructing, 724
fortified line, constructing, 724
fragmentation power domain, 1547
Frame Mode parameter, 232, See also frame-mode parameter
frame mode, changing for fast forward, 261
frame rate, 184, 199, 215-216
exercise clock, 122
mode, specifying, 354
statistics
back-end, 200
enabling and disabling, 1671
troubleshooting, 1660
Frame Rate Monitor Panel, See Performance Monitor Panel
Frame Time parameter, 233, See also frame-time parameter
frame time, changing for fast forward, 261
frameless window, 1598
frame-mode parameter, 144, 261, 354, See also Frame Mode parameter
time management, 143
frame-time parameter, 261, 354, See also Frame Time parameter
free-fly mode, 1215
freehand line, 1012
sampling rate, 1012
Freeze Fuel Levels set data request, 880
Freeze Movement set data request, 880
Freeze Munition Levels set data request, 881
frequency
color of, 1136
radio communication line, 1076
sampling for terrain profile, 1267
front-end, 114
coordinating multiple, 117
feature representation, 1244
joining session at startup, 140
multiple, 114
process ID, 181
selecting database, 228
sending messages, 114
starting, 129
frustum, 1604, 1615
editing, 1609
fsm, module, 786
FTH, 1661
fuel, 537, 873
freeze levels, 880
pacing and tracking, 921
refuel supply, 1391
specifying, 920
specifying amount, 896, 1391
status, 465
fuel tank, dropping, 738
Full Color Mode, 1119
Full Knowledge CID level, 302
full screen, 173
hiding mouse, 1629
mode, 168
window, 1598, 1629
function
check(), 789
checkInit(), 789
endTask(), 796
entityDestroyed(, 789
getParameterProperty(), 1493
getStateProperty(), 1493
init(), 787
isSubtaskCanceled(), 797
isSubtaskRunning(), 797
isTaskCanceled(), 797
isTaskComplete(), 797
isTaskRunning(), 797
list, filtering, 1655
loadState(), 788
makeCopy(), 785
movement, 1650
profiler, 202
receiveObjectMessage(), 788

receiveTextMessage(), 789
resume(), 788
saveState(), 788
setCheckpointMode, 794
shutdown(), 788
stopSubtask(), 797
stopTask(), 797
subtaskResult(), 797
suspend(), 788
taskResult(), 797
tick(), 787

G

gain, 1116
changing, 1201
sensor effects, 1117
Gamepad, 1578
installing on Linux, 1589
Gameware Navigation Lab, 1287
gamewareMemorySize parameter, 1670
gamewareQueryTimeBudget parameter, 1671
GARS coordinate system, 1635
gasoline, 537
GDB terrain, 1243
configuring soil type, 1263
shapefile, 1244
generating
navigation data, 1281-1282
entity for, 1285
traffic, 416
geocentric
converting to geodetic, 1669, 1675
coordinate system, 1246
vector, 803
terrain, 1246
geocentric coordinate system, 1182, 1635
geodetic
converting from geocentric, 1669, 1675
geometric objects, 801
geometry
extruded, 1244
object, adding to entity, 1380
sharing, 204
GeoTIFF
elevation, 1243
imagery, 1243
gesture
task, 654
getParameterProperty() function, 1493
getStateProperty() function, 1493
ghosted icon, by echelon level, 446
ghosted unit, 446
GIF, 1243
gimbaled sensor, 899, 1228
control panel, 1230
create view command, 978
field of view, 1229
setting azimuth, 1232
GL Studio, 90, 1225
instrument panel, 1227-1228
license, 91
global
dynamic terrain processor, 232
environment, 232
weather, 232
global command
adding to plan, 969
Create, 973
Delete Object, 973
Delete Self, 973
Issue Plan, 973
global plan, 932
creating, 966
file, 225
running, 968
global setting, 126
Go To Location Panel, 1207
God ray, 1095
GPS Jammer set data request, 881
GPU performance, 170, 1307
graphic
adding to scenario event, 279
buffer, depth and stencil, 207
depth, 164
fire and detonation, 1058
object, showing and hiding, 998
unit combat, 1059
wireframe mode, 1254
graphical object, 994
deleting, 1029
line width, changing, 1033
spot report, 295
graphical user interface, See GUI
gravity, aligning cultural feature, 1382
grayscale, terrain, 1260
grenade
creating, 1393
hand, 648
types, 648
grid line
configuring, 1253
displaying, 1252
keyword, 1613
grid north, 1199
grid spacing, changing, 1416

GridLines keyword, 1613
Ground Clamp Fidelity set data request, 881
ground clamping, 1248
  configuration, 1249
  configuring, 881
  cultural feature, 1382
ground cover
  rain, 329
  snow, 329
ground movement, 502
  disable autonomous action, 508
ground truth, 290-291, 295
ground vehicle
  changing drag and traction factors by soil type, 506
  configuring obstruction avoidance, 1568
  measurement unit, 1634
  road driving, 503
ground, attacking from air
  aggregate-level, 714
  entity-level, 630
ground-entity-state-repository, 1476
ground-vehicle-param parameter, 1476, 1688
group
  selection, 457
  simulation object, 400
GUI, 114
  configuration file, 168
  configuring, 1640
  layout settings, 168
  layout, saving, 1624
  localizing, 108
  locking, 1647
  modes, light and dark, 1637
  settings
    file, 225
    scenario, 350
  Simulation Object Editor layout settings, 1307
guidance-mode parameter, 1546
GuiObserverViews parameter, 355
gui-runtime-scheme parameter, 354
gui-runtime-scheme-data parameter, 354
GuiScenarioSettings parameter, 355
Gui-Terrain-Database parameter, 353
gumball, 466
gun, attacking aircraft with, 625

H

hail intensity, 328
Halt Movement task, 733
hand grenade, 648
  types, 648
hand item, hiding, 902
hard-freeze-temperature parameter, 507
Has Drag Chute set data request, 921
Hasty-Attack posture, 530
Hasty-Defense posture, 530
hazard area, 532
HBAO
  enabling, 209
  settings, 211
HDR lighting, 1097
header file
  objTypeEnums.h, 1688
  radioMessageTypes.h, 1645
  vrfMenuIds.h, 1646
heading, 470, 1177
  flying to, 591-592
  indicator, 389, 470
  object, changing, 393, 1030
  observer, 1178
  pasting, 397
  rotating icon to, 1082
  setting, 882
    manually, 389
  specifying, 372
  specifying measurement unit, 1634
  turning to, 622, 736
  view with range line, 1130
  zig zag to, 622
Heading set data request, 882
headlights, 1090
headquarters section
  dismount, 669, 909
  mount, 671, 910
health, 528
  restoring simulation object, 896
heap, fault tolerant, 1661
heartbeat
  back-end, configuring, 1675
  DIS, 1678
  relationship to thresholds, 212
  varying, 1679
heartbeatVariance parameter, 1679
heavy lift system, 611
height, See also altitude
  fog, 328
  terrain data, 1243
height-above-terrain line, 480, 998, 1200
  route, 1001
  track history, 1056
  vertex, 1000
helicopter, See also rotary-wing entity
  fast rope insertion, 686
  ladder extraction, 688
  ladder insertion, 690

rope extraction, 686
rope insertion, 688
rope sling load drop off, 685
rope sling load pick up, 685
rotor wash, 1109
terrain avoidance, 514
help
HTML5, 66
help, command-line options, 175, 1151, 1309
help, HTML5, 104
herd
adding members, 652
area, 434
creating, 433
hidden terrain, 1605
HiddenTerrainViews parameter, 355
Hide Occupants, 683
Hide Occupants set data request, 882
hide_by_default, 1489
hiding
3D models, 1084
all panels, 1624
hand item, 902
intervisibility lines, 1129
mouse, 1629
objects, 998, 1006
overlay objects, 1006
panel, 1624
simulation objects, 290
tactical graphics, 1035, 1038
toolbar, 1624
vertex labels, 1000
hierarchy
entity, 364
file
merge, 171
high dynamic range, 1094, 1097
high fidelity damage area, 1561
high-explosive resource, 537
high-fidelity-damage-table, 1561
high-fidelity-terrain-damage-table block, 1561
hit file, 1527
hit probability
configure, 1538
file, 1502
skill level, 1538
table, 1530
HLA
4, 129
1516, 129
troubleshooting, 274
command-line options, 190
federation execution name, specifying, 190
object discovery optimization, 198
parameters, 1677
starting in, 168
startup tutorial, 134
time management, 142
holding pattern, aircraft, 610
horizon-based ambient occlusion
enabling, 209
settings, 211
horizontal navigation, 1650
hose-and-drogue refueling, 701
host address, 175, 177, 1671
hostAddressString parameter, 1671
hostile-fire-only parameter, 521
hostility
changing in a plan, 302
force relationships, 300
matrix, 300
relationships file, 225, 358
hostname
license server, 105
Hover task, 594
HQ, See headquarters section
human
adding to crowd, 652
animation speed, 908
getting in and out of car, 683
idling, 650
ladder extraction, 690
ladder insertion, 691
praying, 650
protesting, 651
releasing from queue, 706
rope extraction, 687
sliding down rope, 686
soldier name tag, 889
soldier rank patch, 894
soldier unit patch, 908
talking, 651
tasks, 649
urgency, 908
waiting in line, 664
human-param parameter, 1476, 1688
human-state-repository, 1476
humidity, 333

I

icon
2D, 381
changing, 1348
customizing, 479
image, 1349

scale, 1080
borders, 1080
changing on objects list panel, 1343
common, 1630
displaying label, 473
heading, rotating to, 1082
level of nesting, 446
light mode and dark mode, 1637
list, changing, 1343
location of, 103
notification, 486
Set menu, 753
simulation object, 1080
size, 1080
spot report
changing, 1351
updating, 1349
tactical graphic, 1364
Task menu, 753
toolbar, specifying, 1646
triangle, 486
warning, 303, 486
yellow, 303, 486
icon list
creating, editing, and removing, 1343
specifying entity icons, 1344
using, 1344
ID
application, 176
echelon, 363-364
emitter, 894
marking, license plate number, 1357
object, 364
script, 750, 753, 768, 796
editing, 753
session, 129, 133, 150, 152, 176-177, 183
site, 156, 176-177, 183, 189, 1150, 1675
task, 796
unique, 363
Identification Friend or Foe, See IFF
identifier
simulation object, UUID, 362
site and application, 115
idling, human, 650
IED, 1536
arming, 870
detonating, 875
placing, 643
If statement, 933, 935
building, 956
in plan, 950
IFF
information, 463
Interrogate task, 693
set data request for IFF transponder, 883
Sweep Interrogate task, 694
ignore-sets parameter, 1645
illumination, 1515
model, 232, 332
setting, 331
image
2D icon, 1349
adding, 1351
list in script, 762
supported formats, terrain, 1243
imagery
filtering, 1255
ImGui overlay
debug, 221
immediate embarkation, 407
impact-point range, 1547
importing
element definitions, 1359
entity file, 1326
model definitions, 1359
MSDL, 238, 241, 1672
navigation area, 1287
navigation data, 1287
object from CSV files, 243
object information, 1327
object tags, 246
object type mappings, 1359
order of battle, 495
scenarios, 238-239
scripted task, 777-778
shapefiles to overlay, 1009
tags, 1331
visual models, 1359
VR-Forces application settings, 123
Improve Booby Trap task, 727
Improve Breach task, 728
Improve Bridge task, 728
Improve Ditch task, 728
Improve Fortification task, 729
Improve Obstacle task, 729
improve-script-id parameter, 768
improving
anti-tank ditch, 728
booby trap, 727
breach, 728
bridge, 728
ditch, 728
fortification, 729
obstacle, 729
strong point, 729
improvised explosive device, See IED

inactive tactical graphic, 1035, 1038
included SMS, 1310
  application-level files, 1314
  hiding objects, 1314
  priority, 1312
increasing speed of navigation, 1200
independent mode, startup tutorial, 134
independent settings, 126
independent task, 554, 558
index, extended label, changing, 478
indicated airspeed, 464, 473
indicator
  entity, 471
  entity, embarked, 472
  heading, 470
  squawk, 1073
indirect fire, 314
  activation, 1035
  deactivation, 1035
  editing, 318
  event
    configuration, 318
    disabling, 1038
    new, 314
  providing, 644
  trajectory, 893
  weapon system, 1534
Indirect Fire task, 718
Indirect Fire, Provide task, 644
indirect rendering
  external, disable, 165
indirect-fire-actuator, 1534
indirect-fire-controller, 1534
indirect-fire-weapon-controller, 1534
influenceName parameter, 789
influenceParameters parameter, 789
influencer parameter, 789
information
  connection, 153
  entity label, 469
  message, 1672
  multiple objects, 468
  object, 462
  scenario, displaying, 252
Information dialog box
  aerodrome, 1294
  centering observer from, 1198
  configuring, 1642
  display user-created state properties, 1493
informationitem element, 1643
informationpagecollection element, 1642
infrared, 1116, 1164
ingress point, 1442, 1569
  configuring, 1451
ingress technique, 1450
ingress-points parameter, 1569
init() function, 787
initialization data, FOM Mapper, 1677
initial-road-preference parameter, 890
initialSetMenuConfiguration.xml, 1645
initialTaskMenuConfiguration.xml, 1644
input
  of data, 1477
  port and port group, 1478
Insert Using Ladder task, 691
inset view, 1224
installation, 96
  data
    options, 101
  default directory, Windows, 97
  documentation, Windows, 97
  Linux, 97
  RTI, 107
  Windows, 96
instancing, DI-Guy, 220
instant embarkation, 407
instrument panel, 1226
  channel keyword, 1613
  configuring, 1228
  GL Studio, 1227-1228
instrumentPanelCategory parameter, 1228
Intel Collection Area, 1020
intelligence object, 281
intensity, precipitation, 328
interaction, 123
  detonation, 1534
  fire, 1532-1533
  signal, 1073
interactive script, 805
intercom, 1148
interface
  Lua, 784
intersection, KD trees, 165
interval
  spawn, 420-421, 424
interval, timeout, 176
intervisibility, 1122
  enabling and disabling, 1129
  entity, 1123
  fan, 1123-1124, 1126
    moving, 1126-1127
    simulation object, 1124
  grid, 1123
  line, 994
    terrain profile, 1267

line, show or hide, 1129
object
configuring, 1127
deleting, 1130
displaying, 1126
editing, 1126
transient and permanent, 1124
pinning object to simulation object, 1128
point-to-point, 1123
terrain profile, 1129
terrain profile line, 1269
types, 1123
invert pattern, 963
inverting condition parameter, 963
Invisible set data request, 884
invisible terrain, 1605
Invulnerable set data request, 884
IP address, 176-177, 1674
IPV6, 152, 177, 184, 1150, 1679
isSubtaskCanceled() function, 797
isSubtaskComplete() function, 797
isSubtaskRunning() function, 797
Issue Plan command, 973
isTaskCanceled() function, 797
isTaskComplete() function, 797
isTaskRunning() function, 797
itemname attribute, 1640, 1646

J

jammer
activating, 737
deactivating, 737
emitter ID, 885
mode, 885
Jammer State set data request, 885
jamming
communication, 539
MAK radio frequency, 1159
radar, 306, 539-540
enabling and disabling, 885
target, 866
Jamming Mode set data request, 885
jamming-defense-factor parameter, 540
jam-strength-factor parameter, 541
jitter, camera, 1198
joining exercise late, 118
joint engagement zone, altitude limited, 1016
joystick, 1578
configuration
creating, 1581
favorite, 1588
configuring, 1579
controller, 1579
mapping, 1582
disabling, 1579
Gamepad, 1589
keyboard contorl, 1579
Joystick button, 1582
JPEG, 1243
JPG, 1243
JRM Technologies, 1116
jumping the observer, 1189
jumping, human posture, 892

K

KD tree, 165
Keep Station task, 595
key map, 1650
creating, 1655
deleting, 1655
editing, 1651
key mapping, 1185
adding, 1651
binary, 1651
changing, 1653
combining, 1654
deleting, 1653
filtering function list, 1655
key value pair, reader writer, 1440
keyboard, 1650
attaching to objects from, 1211
changing speed, 1201
control, 1629
mapping, 1583
controlling entities, 1579
creating selection group from, 459
entity control, 1579
inset views, 1224
mappings, 1181, 1184, 1650
movement functions, 1181
moving observer, 1184
navigation, 1650
key-names, location of configuration files, 1593
keyword
channel, 1604, 1613
cockpitOnly, 1613
GridLines, 1613
largeCompass, 1180, 1613
NoExternalInput, 1613
showCockpits, 1613
showRemoteGraphics, 1613
simulation object property, 1383
smallCompass, 1180, 1613
tactical graphic property, 1365

kill data, 1558
kinetic power domain, 1547
kneeling, human posture, 892

L

label, 362
  2D
    LOD out Altitude, 475
    position, 475
  background, 476
  displaying, 473
  entity, 473
    2D, 470
    3D, 469
    enabling and disabling, 471-472
  extended, 478, 995
  object, 364, 473, 995, 1000
  overlay, 1006
  pinning to display, 472
  showing and hiding, 473
  simulation object, 364, 473
    extended, 477
  spot report, 296
  tactical graphics, 995, 1000
ladder
  extraction, 688, 690
  insertion, 690-691
Ladder Extraction task, 688
Ladder Insertion task, 690
Lambert Conformal Conic coordinate system, 1182
Land (Vertical) task, 598
Land at Air Base task, 734
Land at Airport Near Location task, 596
Land at Airport task, 596
land attack missile, 715
Land on Route task, 596
Land on Runway task, 597
landing
  air base, 734
  fixed-wing entity, 511-512, 596-597
  rotary-wing entity, 612
  vertical, 598
language
  specifying
    for GUI, 189
language, scripted task, 770
language, specifying, 112, 170, 187, 189, 1306
largeCompass keyword, 1180, 1613
Lase and Fire Guided Missile task, 637
Lase Autonomous set data request, 305
lase location, 637
Lase Target task, 304, 637
laser
  bomb, 646
  code, 885
    range, 304
    synchronizing between entities, 305, 903
    using, 304
  designator
    configuring, 1071
    displaying, 1070
  pointer, See laser, designator
  power, 886
  targeting, 637
  wavelength, 886
Laser Code set data request, 304-305, 885
Laser Power and Wavelength set data request, 886
laser-guided missile, 636-637, 1533-1534
lasing
  autonomous, 304
  controller, 1533
Last Clicked Environment Panel, 340
late joiner, 118
latency statistics, back-end, 200
lateral-clearance parameter, 1568
latitude, 801
latitude and longitude
  coordinates, 1635
  entity label, 470
  going to, 1207
Launch Anti-Submarine Missile (Vertical) task, 638
Launch Ballistic Missile at Location task, 639
Launch Ballistic Missile at Target Object task, 639
Launch Countermeasures task, 311
Launch Flight task, 730
Launch QRA Aircraft task, 731
Launch Smoke task, 640
launcher controller, 1533
Launcher, VR-Forces, 129
launching
  countermeasures, 311
  torpedo, 641
layer
  earth file
    hiding, 1255
  feature, 1244
layout
  default, clearing, 1626
  formation
    automatic, 1415
    manual, 1415
  GUI, 168, 1307
    choosing, 1625

creating, 1624
default, 1626
deleting, 1628
renaming, 1627
restoring factory, 1628
reverting, 1627
saving, 1624
updating, 1627
LCC coordinate system, 1182
leader-promotion-id parameter, 1566
length
intervisibility object, 1126
simulation object name, 363
track history, 1055
level observer, 1182
level of notification for messages, 189
level-of-detail, 1191
library
dynamic, 157
FOM Mapper, 175, 185, 191-192, 1151, 1677
license
3D models, 92
DI-Guy, 89
GL Studio, 91
OpenSceneGraph, 93
osgEarth, 93
SilverLining, 90
SpeedTree, 92
license manager, 104
license server
hostname, 105
lifeform, 941
appearance, random, 1385
configuring DI-Guy appearance and character, 1383
DI-Guy appearance and character, 1464
measurement unit, 1634
posture, 892
Lifeform Surrendered condition, 941
lifetime
radio communication line, 1076
squawk indicator and communication line, 1076
light
daylight control, 1092
flashlight mode, 1097
point, 1092
source, 1092
Light (Advanced) set data request, 1295
light mode, enabling and disabling, 1637
lighting
aerodrome, 1295
aircraft, 868
tanker, 904
aircraft, preview, 1353
disabling, 1100
dynamic, 1090
effects, 86
high dynamic range, 1097
lens flare, 1094
vehicle, preview, 1353
watercraft, 903
watercraft, preview, 1353
limited munition attack, 532
line
arrow, editing, 1033
changing width, 1033
creating
displaying terrain profile, 1018
opening terrain profile, 1267
direction, changing, 1033
dotted and dashed, 1033
editing, 1033
opening terrain profile, 1267
fire, 1058
fortified, constructing, 724
freehand, 1012
height-above-terrain, 480, 998
intervisibility, 994
phase, 996
radio communication, 1073
range, 1132
sensor contact, 1063, 1065
style, 1033
supply, 1071
suppressive fire at, 520, 637, 645
task visualization, 1060, 1062
terrain profile, 994, 1267, 1269
for intervisibility, 1129
vertex, showing in terrain profile, 1267
waiting in, 664
Line Width set data request, 1040
linear feature list, 1258
linear obstacle, 994, 997
linear scaling, 1082
line-of-sight, See also intervisibility
communication line, 1076
terrain profile, 1129
unit versus entity, 533
linked event, 280
Linux
installing Gamepad, 1589
product installation, 97
troubleshooting, 1661
lisp, 1682
list
clearing Countries, 1342
Countries, 1342

image in script, 762
of components in OPD, 1481
spot report, 296
target priority
add to, 905-906
clear, 906
update, 906
viewing, 905
list box, multiple, 563
load balancing, 197, 255
scenario, 233
delete simulation object, 232
existing, 255
new, 255
on the fly, 257
rule set, 256
loadAllNavigationDataOnTerrainLoad parameter, 1671
load-balanced scenario
saving, 248
starting, 258
load-entry parameter, 1569
load-entry-point parameter, 1569
loading
database, 1246
display engine configuration, 165, 1602
entities on entities, 406
entity display settings, 166
environment configuration, 167
object type mapping, 167
plug-ins, 157, 170, 173, 181, 190
saved views, 1206
scenario, 173, 182, 249-251
recently used, 251
scripts, 787
settings, 123
tactical graphics, 1007
terrain, 144
from command line, 174
paged, 181
Loadout set data request, 886, 922
loadPluginIfVersionMismatch parameter, 1671
loadState() function, 788
local
object, 120, 1473
network interface, 1477
simulation object, 120
weather, 336-338
locale, specifying, 112, 170, 187, 189, 1306
localization
specifying, 112, 170, 187, 189, 1306
VR-Forces GUI, 108
local-objects parameter, 1473, 1473, 1683
localSettingsDesignator parameter, 1677
location
compass, 1181
fixed-wing at creation, 510
going to, 1207
jumping to, 1189
moving to, 604
new entity, 386
object, repositioning, 1030
observer, 1177
ordering entity to, 604
ordering simulation object to, 604
setting, 887
by dragging, 392
simulation object, repositioning, 393
specifying, 372
suppressive fire at, 520
targeting with laser, 637
window, 1603
Location set data request, 887
Location3D class, 801
locator, embarked entity, 472
locator, entity, 471
Lock Embarkation set data request, 910
locking
GUI element, 1647
overlay, 1005
LOD, 1191
buoyancy, 1111
center, 1195
ocean planar reflections, 1093
terrain
magnification angle, 1193
scale factor, 1192
zooming in, 1193
LOD center settings, 1195
log file, 162, 180, 1670
callstack, 98
location, 99
messages sent to, 189
notification level, 173, 182, 189, 1149, 1308, 1672
notify level, 181, 1149
sending entity messages, 485
specifying, 170, 181, 1149
timestamp, 166
vrfSim, 182
log, database, 88
logFrameRateStatistics parameter, 1671
Logger
recording to, 274
recording view control messages, 975
Logger Control message, 273
logger-files-path parameter, 271

logical operator, 942
logistics, 537
  resupply modes, 925
  supplying, 926
logo, marking on entity, 1357
loiter munition, 642
loiter pattern, flying, 592
longitude, 801
longitude and latitude
  coordinates, 1635
  entity label, 470
  going to, 1207
Look At set data request, 888
loop in plan, 938
Lower Periscope task, 702
Lua, 745, 747
  API Documentation, 104, 784, 796
  background process, 812
  classes, 784
  direction vector, 801
  global variables, saving, 794
  IDE, 843
  interface, 784
  locations and vectors, 801
  method, 784
  module, 786
  object, 784
    location of, 801
  parameter, 797
    as table, 796
  require, 787
  script
    editing, 782
    extended name, 754
  table, 796
  tasks and subtasks, 795
  utility functions, 787
  vector
    geocentric, 803
    offset, 803
  ZeroBrane Studio, 843
lubricant, 537
  pacing and tracking, 921
  specifying, 920
LWO file, 1243

M

Mach number, 464, 473
magnetic north, 1199
magnification, performance, 1193
mainmenu element, 1640
mainmenuconfiguration element, 1640
MAK data
  install
    options, 101
MAK Data Logger, recording to, 274
MAK Earth terrain, 1238
MAK radio, 1148, 1151
  configuring, 1154
  dialog box, 1151
  intercom, 1154
  jamming, 1159
  standalone application, 1151
  toolbar, 1153
  using, 1148
  VOX, 1154
MAK RTI configuration, 107
MAK Technologies LISP, 1664
MAK_INSTALL_DIR environment variable, 97
makeCopy() function, 785
making decisions, 366
MAKLMGRD_LICENSE_FILE environment variable, 105
managing plug-ins, 156
Maneuver Along task, 598
maneuver and fire task, 657
Maneuver To task, 599
manual, location of, 104
map
  color, 1260
  cube, recalculating, 1096
  Sensor Texture, 1117
  texture, 86
map datum, configuring, 1676
map scale overlay, 1263
mapping
  joystick controls, 1582
  joystick keyboard controls, 1583
  keyboard, 1185
  keys, 1650
  model
    to remote observer, 1172
  movement to keyboard, 1184
  objects to back-ends, 258
  overwrite on merge, 172-173
  simulation objects, 248
  sound effects, 1145
marine
  conditions, 343
  effects, 342
marker
  displaying, 1353
  label, displaying, 1353
  system, 1392

marking
  remote simulation object, 945
  specifying, 888, 1357
  visual, 888, 1357
marking text, 470, 1340
marksmanship, 900
matching object type, 1470
material, ambient value, 1089
matrix
  hostility, 300
  synchronization, 984
maxAsynchronousTerrainThreads parameter, 1671
max-detonation-distance-for-passing-rounds parameter, 521
maxDrainInputReads parameter, 1679
maxDrainInputTime parameter, 1672
maximum sheaf parameter, 1426
Maximum Skill Level Effect parameter, 1538
maxNavigationRegenerationThreads parameter, 1672
max-range parameter, 646
max-range-altitude parameter, 646
max-slope parameter, 512
max-speed parameter, 891
max-thrust parameter, 512
mcastTtl parameter, 1679
MEDF file, 1243
media, adding to scenario event, 279
memory, 199
  GPU usage, 170, 1307
  improving use of, 204
menu
  configuring, 1640
  context-sensitive, 1630
  locking, 1647
  separator, 1640
  Set
    editing, 1645
    sorting, 1640
  shortcut, 1643
  Task
    editing, 1644
    sorting, 1640
  tearing off, 1628
menuitem element, 1644
merge-with-sub-systems parameter, 1524
merging scenarios, 239
mesh, navigation, 1023
message
  entity, 483
    saving, 485
  frequency of, 1670
  information and warning, 1672
  notification level, 173, 182, 189, 1149, 1308, 1672
  notification of, 486
  object console
    sending, 972
    viewing, 484
  radio, 695-696
  receiving, 1574
  sending, 1574
    between front-end and back-end, 114
    text, 972
    text (command), 972
    text (task), 697
  sent, 367
  session, configuring, 140
  setting levels, 189
  Start/Resume, 273
  state update, 1172
  Stop/Freeze, 273
  tactical data link, 298
  TDL, 298
  timestamp ordered, 143
  view control, 1207-1208
  warning icon, 486
metadata
  in system definition file, 1482
  script, 747
meta-data-entry-list parameters, 505
method, Lua, 784
METOC FOM
  object classes, 348
MGRS coordinates, 1635
  going to, 1207
middle-click menu, 1630
MIL
  bearing and range, 1634
  compass, 1179, 1634
military marking, 1357
Military Scenario Definition Language
  exporting, 241
  importing, 241
MIL-STD 2525B color, 479
  unit, 450
Mimic Location Track mode, 1217
Mimic observer mode, 1182, 1214-1215
Mimic Track observer mode, 1216
mimModule parameter, 1677
minDrainInputTime parameter, 1677
mine
  creating, 1393
  dropping, 626, 628
  minefield, creating
    aggregate-level, 725

entity-level, 1021
naval, 624
sweeping for, 619
minefield
aggregate-level, 725
aggregate-level modeling, 535
entity-level, 1021
minimal-illumination-value parameter, 1516
minimum lift speed, 512
minimum tick period, 206
minimum tick period variance, 206
Mirror Mode, 1604, 1612
missile, 1533
anti-air, 715
anti-ship, 715
anti-submarine, 638
attack, 532
autonomous, 642
ballistic, 319, 639
firing, 320
target event, 321-322
controller, 1533-1534
creating, 1393
cruise, 633
fixed trajectory, 319
land attack, 715
laser-guided, 636-637, 1533-1534
launcher, See missile launcher
loadout, 886
strategic, 319
target event, 321-322
types, 1532
Missile Approach Warning condition, 941
missile engagement zone
altitude limited, 1016
missile launcher, 1532
laser-guided, 1533
system, 1532
missile-collision actuator, 1533-1534
missile-entity-state-repository, 1476
missile-maneuver actuator, 1533-1534
missile-param parameter, 1476, 1688
Mission Oriented Protective Posture. See MOPP, 532
mobility modifier, 533
mode
Absolute, 1215
batch, 269
C and S / 5 in IFF, 883
flashlight, 1097
Follow, 1215
jammer, 885
Mimic, 1215
Mimic Track, 1216
observer, 1164
Space Follow, 1217
specifying, 192
speed scaling, 1201
stealth, 907
Tether, 1218
Tether Track, 1219
Track, 1219
view, 1182, 1214
model
3D, 381
changing, 1347
DI-Guy, 654
exaggerating entity, 1082
hiding, 1084
scaling, 1082
aggregate-level, creating, 1418
calculating support point from, 1379
colorized, changing, 1347
constructive, See aggregate-level
decal effects, 1052
definition
prevent loading, 172
DI-Guy, choosing, 1466
entity, 1309
editing, 1305, 1337
illumination, 332
instancing cache, 204
layer
disabling, 1662
filtering, 1255
mapping
common buttons, 1630
to remote observer, 1172
name, 470
object, 1309
physical, 367
preloading, 218
file, 219
list, 219
set, 1172
changing, 1353
displaying, 1352
sharing resources, 204
simulation, 1309
stealth, 1173
tactical graphic, 1366
topological, 123
trailing effects, 1052
type, changing, 1356
vehicle, 79

visual
adding, 1351
editing, 1346
refreshing, 1360
wake configuration, 1395
model definition
adding, 1351-1352
compass, 1180
DefaultEllipsoidArc, 1136
exporting, 1359
importing, 1359
merging, 172
pruning, 1359
tactical graphic texture, 1368
wake, 345
model set, disable, 165
modeling, aggregate- and entity-level, 368
modifier, 528, 1430
combat engineering object, 533
mobility, 533
visibility, 535
vulnerability, 534
modifying
control object, 1029
sensor signature, 1020
module
fsm, 786
location of, 787
Lua, 786
monitor
multiple, 1613
performance, 217
MOPP level, 532
changing (task), 732
MOPP Level set data request, 922
Morale set data request, 923
motion
file, 1465
Mount Headquarters task, 671
Mount Platoon task, 671
Mount Squad task, 672
Mount Team task, 672
Mounted Crew set data request, 910
Mounted Headquarters Section set data request, 910
mouse
changing speed, 1201
drag terrain, 1187
drag view, 1189
hiding in full screen mode, 1629
locking to object, 390
moving observer, 1187, 1189, 1191
orbiting terrain, 1189, 1191
teleporting observer, 1189
Move Along Route Retrograde task, 734
Move Along Route task, 599
functionality, 505
Move Along Route with Actions task, 600
Move Between Cover to Location task, 659
Move Into Formation task, 603
Move Next to Target task, 603
Move task, 734
Move To (Plan Path) task, 606
Move to Altitude task, 605
Move to Bullseye Offset task, 605
Move to Depth task, 568, 606
Move to Location (Plan Along Roads) task, 735
Move to Location task, 604
Move to Retrograde task, 735
Move To task
functionality, 500
in plan, 604
path planning, 500
Move to Waypoint (Plan Along Roads) task, 735
movement
DI-Guy, 654
freezing, 880
ground vehicle, 502
lifeform, 892
model, 1390
specifying, 1392
obstacle avoidance, 502
retreat, 734-735
scripted, 1401
speed scaling, 1200
speed, affect of posture, 530
stopping, 733
system, 1391, 1482
properties, 1433
unit, 517, 531
movement file, configuring, 1403
movement parameter, 1432
Movement To Contact task, 677
moving
along a route, 599
along a route with actions, 600
articulated parts, 698
camera, 1186
in formation, 945
intervisibility fan, 1126-1127
object, 391-392, 887
between overlays, 1017
observer, 1177, 1185
mouse with, 1187, 1189, 1191
options, 1185
to altitude, 605

to another entity, 603
to location, 604
vertex, 1031
moving object parameter, 1687
moving-object-param parameter, 1477, 1688
moving-object-param parameter type, 1687
MSDL
exporting, 241
importing, 238, 241, 1672
MTL file, 1664
vrfSim.mtl, 1668
multicast
address
specifying, 177, 187, 192, 1151
address, configuring, 1678
mode, specifying, 192
time-to-live, 193, 1679
multicast TTL
setting, 177, 182, 1149, 1679
multichannel
configuration, 1613
seamless, 1661
multimedia, adding to scenario event, 279
multiple
back-ends, 114, 118, 357
front-ends, 114
coordinating, 117
objects, information, 468
simulation model sets, 233
viewports, 1610
VR-Forces processes, 114
multiple hit damage system, 522
damage percentage, 875
multiple list box, 563
multiprocessor usage, 197
multiscreen performance, 217
multithreading
RTI issues, 1670
munition
configure, 1529
display of path and detonation, 1058
freeze levels, 881
loiter, 642
non-simulated, 1393
creating, 1394
setting supply for an air base, 923
simulated, 1393
creating, 1393
previewing, 1394
tracer, 1394
munition block, 1546
munition-type parameter, 1529
mutually exclusive task, 554, 561, 935
configuring, 580
mutually-exclusive-task-groups parameter, 580
muzzle flashes, 1057

N

name, 362
changing
simulation object, 1340
tactical graphic, 393
component, 1478
control object, 994
default, 382
displaying, 470
entity, 469-470
extended, 754
FED file, 1677
federate, 1677
in condition, 963-964
object, 1340
conflicts in merged scenarios, 239
editing, 394
short, 1340
simulation object, 363, 469, 1340
slot, 682, 1442
specifying, 372
trigger, 936
Nametag set data request, 889
NATO-MIL
bearing and range, 1634
compass, 1179, 1634
nav edge, 1023
Naval Depth Charge Deployment system, 625-626
Naval Mine Deployment system, 626, 628
naval mine explosive device system, 1536
nav-edges, 1282
navigation
aid, 904
attached context, 1182
function, editing key map, 1651
key map, 1650
keyboard, 1650
mesh, 1023
observer view, 1177
options to, 1185
planning, performance, 247
speed scaling, 1200
tactical air, 904
navigation area
canceling generation of navigation data, 1283
clamping to, 881
configuration file, 1287

creating, 1277
disabling use of, 889
editing, 1280
reverting, 1280
generating navigation data for, 1282
importing, 1287
information, 1280
nav edge, 1023
pedestrian path, 435, 994
removing, 1280
sector, 1277
status, 1283
navigation data, 1272
displaying, 1287
generating, 1281-1282
canceling, 1284
entity, 1285
removing from queue, 1283
runtime, 1277
importing, 1287
profiles, 1281
regenerating, 1672-1673
automatically, 1284
troubleshooting, 1672
using, 501, 1284
Navigation Enabled set data request, 889
Navigation Lab, 1287
debugging, 1670
viewing simulation objects in, 1289
Navigation Preferences set data request, 890
navigation-preference parameter, 890
navigationProfiles.mtl
regenerating navigation data, 1284
navigationRegenerationDelay parameter, 1672
navigationRegenerationFileDirectory parameter, 1672
NBC
area, 532
report, 739
sending, 739
near clipping plane, 1605
Near Far Clip Policy attribute, 1606
net-interface-min-tick-period parameter, 1685, 1687
net-interface-min-tick-period-variance parameter, 1685, 1687
NETN callsign, 873
NETN FOM
revision, 151, 186, 1149
version, 151, 186, 1149
network
connecting to
automatically, 164
vector, 503
network interface, 1685
hierarchy, 1477
network-interface parameter, 1685, 1687
new window, 137
night vision, 1116
nighttime-illumination-value parameter, 1516
no dialog box, 188
NoExternalInput keyword, 1613
noise, 1117
adding to scenario event, 279
level, 1142
propulsion, RPM, 310
sound effects, 1142
non-simulated munition, 1393
non-VR-Forces simulation object, 945
normal map, 86
north
grid, 1199
magnetic, 1199
orienting Plan View, 1199
true, 1199
notation prefix, 956
notification icon, 486
notification level, 1672
setting, 173, 182, 189, 890, 1149, 1308, 1672
entity console, 483
Notify Level set data request, 890
tactical graphics, 1041
notifyLevel parameter, 1672
NTC (online) terrain, 1239
Nuclear Attack task, 718
nuclear contamination, 532, 718
numbering events, 287
numberOfNavQueueProcessingThreads parameter, 1673

-

--numberOfRemoteNetworkThreads, 177
number-of-runs parameter, 271
numCallbackThreads parameter, 1673
numNetworkCallbackThreads parameter, 1673
NVG, 1116, 1164

O

OBJ file, 1243
object
adding
to Props Palette, 402
to selection group, 459-460

adding to Order of Battle, 491
adding to order of battle, 493
altitude, changing, 393, 1030
attaching observer to, 1210
avoidance, configuring, 1568
changing
  line width, 1033
  name, 393
clutter
  aircraft, 428
  area, 428
  promoting, 893
  ship, 429
combat engineering, 533
concepts, 119
control, 994
control object, editing, 1029
copying, 395-396
count, 199, 487
creating, 372-374, 377, 381-382
  from OOB, 493
  in plan, 973
  mouse lock, 390
creating new in Simulation Object Editor, 1360
creation, enabling and disabling, 1341
cultural feature, 400, 1377
deleting, 1362
  from plan, 973
  from simulation model set, 1362
detaching observer from, 1213
disembarking, 877
dragging, 392
  to new location, 887
embarking in Embarkation View, 408
entity enumeration, editing, 1338
exaggerating models, 1082
fill style, 1033
filtering list of, 456
geometric, 801
heading
  changing, 1030
heading, changing, 393
hiding, 290, 1006
ID, 362, 364
intelligence, 281
intervisibility, 1124
  pinning to simulation object, 1128
label, 364, 473, 995, 1000
list, 357, 1326
local, 120
location
  changing, 1030
location, changing, 393
locking mouse to, 390
Lua, 784
messages, viewing, 484
model, 1309
multiple, information, 468
name, 1340
  editing, 394
orbiting, 1186
order, changing in OOB, 495
overlay, 994
palette, 373
parameter, 1469, 1682
  default value, 358
  editing, 1337
pasting, 395-397
  altitude, 398
placing, 382
properties
  specifying, 372, 384
remapping back-ends, 258
remote, 120
removing from OOB, 494
repositioning, 391-392
selecting, 1337
  for creation, 373
  in Objects List Panel, 456
selecting in multiple list box, 563
setting icon for icon list, 1344
showing and hiding, 998
subcomponents, 1473
super-type, 1470
tag
  changing, 1337
  loading, 246
timeout interval, 176
timeout threshold, 1678
type, 1469
  changing, 1338
  enumeration, 1688
  mapping, See object type mapping
  published and matching, 1470
unselecting, 457
vertex
  adding, 1030
  deleting, 1032
  editing, 1031
views of, 452
VR-Forces, 120
object console, 483
clearing, 487
notification of messages, 486
setting notify level, 483
warning icon, 486

Object Console Summary Panel, 484, 805
  answering questions, 485
  selecting simulation object, 456
object creation palette
  changing icons, 1343
  filtering, 377
object geometry, 401
  adding, 1380
Object Group tag, 400
object information
  exporting, 1326-1327
  file, 1326, 1331
  importing, 1326-1327
object map file, 225, 350, 357
  missing back-ends, 119
object parameter database, 119, 1309, 1468
  component list order, 1481
  configuring
    sensors, 1504
  configuring path to, 1669
  pathname interpretation, 1502
  reloading, 1670
  system definition file, 1482
Object Parameter Database Editor, See OPD Editor
object tick, back-end statistics, 200
object type mapping
  exporting, 1359
  importing, 1359
  loading, 167
  pruning, 1359
objectConsoleNotifyLevel parameter, 483, 1673
Object-Map parameter, 353
Objects List Panel, 440, 452
  changing icons, 1343
  selecting object in, 456
Objects List View
  adding object to unit, 440
  removing object from unit, 441
Objects View, 452
object-type parameter, 1683
object-types-to-spot-report parameter, 293
objTypeEnums.h header file, 1688
obscurants, See visibility
observer, 1177
  adding, 1167-1168
  attaching
    articulated part, 1212
    to simulation object or prop, 1210
    to simulation objects, 1214
  attachment
    in 2D, 1220
    order, 1211
camera, 1186
  changing, 1615
camera shake, 1198
centering on object, 1198
centering on object from Object Console Summary Panel, 1197
compass, 1178
coordinate system
  2D, 1182
  3D, 1182
deleting, 1170
detaching from object, 1213
dragging
  terrain, 1187
  view, 1189
go to location, 1207
heading, 1178
inset view, in, 1224
location and orientation, 1177
mode, 1164
  changing, 1165
  creating, 1166
  deleting, 1167
  editing, 1167
  renaming, 1167
  setting, 126
  switching projection, 1164
  view control message, 975
  XR, 1171
movement, 1214
  functions, 1181, 1650
  speed, 1201
moving
  keyboard with, 1184
  mouse with, 1187, 1191
north, 1199
orbiting, 1186
orientation, 1189, 1199
rain and wave effects, 1106
remote, 1172
removing, 1167
saving view, 1202
selecting, 1167
teleporting, 1189
view
  attachment change, 1211
  disable, 165
  saving, 1202
  transitioning between, 1204
visibility, 333
  to other applications, 1172
ways to move, 1185

zooming, 1191
2D, 1192
3D, 1191
entity extents, 1194
zooming to extents, 1194
Observer Name, 1604
Observer Views Panel
loading saved views, 1206
saving views, 1206
obstacle, 994, 997
avoidance, 502
breaching, 719
configuring avoidance of, 1568
destroying, 727
fill style, 1033
improving, 729
linear, 994, 997
report, 739
sending, 739
obstructed terrain
viewing in terrain profile, 1129
obstruction, configuring avoidance of, 1568
occlusion
horizon-based ambient
enabling, 209
settings, 211
occupancy-director-controller, role in embarkation, 1569
ocean
choppiness, 344
configuration, Triton.config, 1194
depth offset, 344
dynamic, enabling and disabling, 342
effect, 341
LOD, 1194
planar reflection, 1092
configuring, 1093
performance, 1093
SpeedTree, 1093
texture height, 1093
texture width, 1093
quality, 346
shadows, 1100
spray, 1106
swell, 344
synchronization, 168, 323, 342
Triton SDK, 91
wave fetch, 345
zoom, display issues, 1194
Ocean LOD Elevation, 1612
offset
burst, 421
start time, 424
oil, 537
pacing and tracking, 921
specifying, 920
on-ground-tolerance parameter, 512
OOB, 490, See also Order of Battle
opacity, label background, 476
OPD Editor, 1468
OPD, reloading, 1670
OPE file, 1502, 1520
adding state properties, 1490
Open Cargo Door task, 699
Open Door task, 660
Open Window task, 660
OpenFlight
file, 1243
format, 79
opening
car door, 683, 698
cargo door, 699
door, building, 660
inset view, 1224
scenario, 264
simulation model set, 1323
terrain, 1246
window, 660
OpenSceneGraph, 76, 90
environment variables for anaglyphic stereo, 1617
license, 93
operator
boolean, 942
comparison, 942
logical, 942
option
command-line, 164
navigation, 1185
ORBAT, See also Order of Battle
overview, 490
palette, 494
panel, 490
Orbat parameter, 353
orbit
observer, 1186
terrain with mouse, 1189, 1191
Orbit Object task, 607
Orbit set data request, 890
Orbit task, 607
orbital element, 246
Order of Battle, 225, 490
adding simulation objects to, 491, 493
changing order of objects, 495
creating units, 493
exporting, 495
importing, 495

instantiating members in scenario, 493
object, editing, 494
removing objects from, 494
reorganizing, 495
Order of Battle Palette, 494
Order of Battle Panel, 490
Order of Battle View, 452
Ordered Formation set data request, 911
Ordered Speed set data request, 568, 891
Order-Of-Battle parameter, 353
ordnance, unexploded, destroying, 727
organization
entity of, 364
unit, 516
orient north, 1199
orientation, 1177
camera, 1615
clamping, 1249
displaying, 470
icon, 1082
measurement unit, 1634
north, 1199
observer, 1177, 1187, 1189, 1191, 1214
Plan View, north, 1199
orthagonal projection, 1262
osgEarth, 90
license, 93
osgearth_cache, 90
outline
emitter volume segment, 1139
segment, 1139
output port group, 1478
out-the-window view, 126, 1164-1165, 1613
overlay, 85
concepts, 994
custom name, 1341
default, specifying, 1341
deleting, 1007
enabling and disabling, 1005
export to shapefile, 1008
file, 350, 1007, 1019
hidden objects, 1006
hiding, 1006
hiding graphics on, 999
importing shapefile to, 1009
introduction to, 1004
label, 1006
locking and unlocking, 1005
map scale, 1263
moving objects between, 1017
new, 1005
object, 994
specifying, 372
tab, 1006
text, adding, 1017
overlay object
copying, 395-396
moving to different overlay, 1017
pasting, 395-397
spot report, 295
Overlay parameter, 353
Overlays View, 452
overriding
default scenario parameters, 230
plan, 982
overwatch, bounding, 657
ownership transfer, 257

P

pacing
ammunition, 915
equipment, 919
resource, 925
weapon, 927
package, scripted task, 777
pagecollection element, 1641
pagecollections element, 1641
paged LOD
magnification high LOD angle, 1193
magnification scale factor, 1192
paged terrain
loading, 181
paging
terrain, 1247
palette
Create Object, 374
Entity, 372
object creation, 373
changing icon, 1343
image, 373, 376, 1362
Order of Battle, 494
Props, 372
simulation objects, 374
tactical graphics, 374
pan view, 1189
panel
displaying and hiding, 1624
displaying hidden, 1624
dockable, 1622
docking, 1622
floating, 1622
hiding all, 1624
Object Console Summary, 484
tabbed, 1622
undocking, 1622

paper map, 1262
parachute, 660
  using Airdrop from Aircraft task, 684
Parachute to Target task, 660
parachuting, human posture, 892
parameter
  acceleration-factor, 507
  acceleration-to-rpm-factor, 311
  adding in Qt Designer, 854
  additional-algorithm-config, 1561
  additionalDestinationAddresses, 1678
  additional-opds, 354
  additionalSystemScriptsDirectory, 1668
  addMulticastAddr, 1678
  aggregate-formation-controller, 1566
  aggregate-level entity, 1430
  aggregate-level unit, 1419
  aggregate-object-param, 1477, 1501
  aggregate-param, 1688
  all-tasks-exclusive, 580
  angle-of-incidence, 1551
  angleSegmentMax, 1139
  appIdRange, 1678
  appNumber, 1668
  assertOnBlockingTerrainCalls, 1247, 1668
  Attach Components to Remote Entities On, 230
  attack
    editing, 1435
  auto-reorganize, 353
  ballistic-missile-param, 1476
  batchScenarioFileName, 1668
  blockOnAsynchronousOperations, 1669
  bomb-param, 1476
  broadcast, 293
  can-be-radar-jammed, 540
  can-pivot, 622
  cgfDispactherReceivePort, 1669
  clockMode, 1669
  commenting out, 1664
  comm-model-name, 1574
  Component Attachment Table, 232
  component list order, 1481
  Component-Attachment, 353
  Component-Attachment-Table, 353
  component-manager, 1685, 1687
  countryCodesMappingFile, 1669, 1672
  Create_Global_Dynamic_Terrain, 355
  Create_Global_Environment, 355
  create-script-id, 768
  cultural-feature-param, 1476, 1688
  daemon, 1669
  datumShiftX, 1669
  datumShiftY, 1669
  datumShiftZ, 1669
  deep-snow-depth, 507
  defaultHeartbeatThreshold, 1678
  defaultObjectTimeout, 1678
  DefaultObserverView, 355
  defaultParameterDatabasePath, 356, 1669
  defaultTerrainDatabasePath, 356, 1670
  defense, 1436
  delay, 1561
  destAddrString, 1678
  destroyFedExec, 1677
  destroy-script-id, 768
  detection-types, 1528
  detonation-munition, 1561
  detonation-on-destroy, 1561
  deviceAddress, 1678
  diGuyAnimationsFile, 1670
  DIS, 1678
  disabled-by-suppression, 522
  disableParallelPublisherTick, 1670
  disableParallelTick, 1670
  dispersal-angle, 629
  Display Terrain, 231
  disPort, 1679
  doNotUseConsole, 1670
  dr-allow-gui-overrides, 213
  dynamic-terrain-type, 1561
  echelon-level, 1501
  editing in Simulation Object Editor, 1380
  egress-points, 1569
  electronic warfare, 1437
  embarkation-slot-entry, 1569
  embarkation-slots, 1569
  enableLogFileTimestamps, 1670
  enableNavigationLabDebugging, 1280, 1670
  entity-range, 1530
  entity-type, 1530
  entry, 1682
  entVrfDataTimeThreshold, 1670
  equipment, 1438
  execName, 1677
  exerciseId, 1679
  exercise-start-time, 354
  expectedLocalEntityCount, 1677
  federateName, 1677
  federateType, 1677
  fedFileName, 1677
  fixed-wing-entity-param, 1688
  fomMapperInitData, 1677
  fomMapperLib, 1677
  forceParameterDbReload, 1670
  Frame Mode, 232
  Frame Time, 233

frame-mode, 144, 354
  time management, 143
frame-time, 354
gamewareMemorySize, 1670
gamewareQueryTimeBudget, 1671
ground-vehicle-param, 1688
guidance-mode, 1546
GuiObserverViews, 355
gui-runtime-scheme, 354
gui-runtime-scheme-data, 354
GuiScenarioSettings, 355
Gui-Terrain-Database, 353
hard-freeze-temperature, 507
heartbeatVariance, 1679
HiddenTerrainViews, 355
HLA, 1677
hostAddressString, 1671
hostile-fire-only, 521
human-param, 1476, 1688
ignore-sets, 1645
improve-script-id, 768
influenceName, 789
influenceParameters, 789
influencer, 789
ingress-points, 1569
initial-road-preference, 890
instrumentPanelCategory, 1228
jamming-defense-factor, 540
jam-strength-factor, 541
lateral-clearance, 1568
leader-promotion-id, 1566
loadAllNavigationDataOnTerrainLoad, 1671
load-entry, 1569
load-entry-point, 1569
loadPluginIfVersionMismatch, 1671
local-objects, 1473, 1683
localSettingsDesignator, 1677
logFrameRateStatistics, 1671
logger-files-path, 271
Lua table, 796
maxAsynchronousTerrainThreads, 1671
max-detonation-distance-for-passing-rounds, 521
maxDrainInputReads, 1679
maxDrainInputTime, 1672
maximum sheaf, 1426
Maximum Skill Level Effect, 1538
maxNavigationRegenerationThreads, 1672
max-range, 646
max-range-altitude, 646
max-slope, 512
max-speed, 891
max-thrust, 512
mcastTtl, 1679
merge-with-sub-systems, 1524
meta-data-entry-list, 505
mimModule, 1677
minDrainInputTime, 1677
minimal-illumination-value, 1516
missile-param, 1688
movement, 1432
moving-object-param, 1688
mutually-exclusive-task-groups, 580
navigation-preference, 890
navigationRegenerationDelay, 1672
navigationRegenerationFileDirectory, 1672
net-interface-min-tick-period, 1685, 1687
net-interface-min-tick-period-variance, 1685, 1687
network-interface, 1685, 1687
nighttime-illumination-value, 1516
notifyLevel, 1672
numberOfNavQueueProcessingThreads, 1673
number-of-runs, 271
numCallbackThreads, 1673
numNetworkCallbackThreads, 1673
object, 1469, 1682
  editing, 1337
objectConsoleNotifyLevel, 483, 1673
Object-Map, 353
object-type, 1683
object-types-to-spot-report, 293
on-ground-tolerance, 512
Orbat, 353
Order-Of-Battle, 353
Overlay, 353
parameter-type, 1469, 1472, 1476, 1501, 1682-1683
passing-rounds-suppress, 521
personnel, 1438
physical, 1431
Plan, 353
plan-manager, 1685, 1687
pluginsDirectory, 1673
position-offset, 1566
power-plant-active, 310
power-type, 1561
promotion-id, 1566
providing information to components, 366
publish-transmitter, 1575
publish-transmitters, 1574
publishZeroVelocityWhenPaused, 1673
Radio Button Choice, 762, 857
rain-to-attenuation-coefficient, 1516
Random Number Seed, 232
random-number-seed, 353
range-determinant, 1530
range-from, 1547
range-list, 1530

recvBufferSize, 1679
regenerateNavigationDataAtRuntime, 1673
rejectMismatchedScenarioMessages, 1673
relative-heading, 412
relative-position, 412
remote-attachment-scheme, 354
remote-attachment-scheme-data, 354
remote-objects, 1473, 1685
reportedHostAddressString, 1674
report-resend-period, 294
reuseEntityIdentifiersOnScenarioLoad, 1674
rotary-wing-entity-param, 1688
round-down, 1552
rprFomVersion, 1678
RTI_timeMgmt, 143
RTI_useRtiExec, 143
run-duration, 272
run-duration-time, 354
runInTimeManagementMode, 144, 1678
saturation-water-depth, 507
scenario
  file, 351
  setting, 230
Scenario Name, 230
scenario-data, 355
ScenarioDescription, 355
ScenarioExtentInformation, 355
Scenario-Extras, 353
scenarioFileName, 1674
scenario-filename, 271
scenario-name, 354
Scenario-Scripts, 353
script, 755, 758
scripted task, list of, 800
script-ids, 768
SelectionGroups, 353
sendBackendLogToNetwork, 1674
sendBufferSize, 1679
sendFedTime, 1678
sendMessagesInNetworkProcess, 1674
send-only-to-front-end, 293
send-spot-reports-on-own-force, 293
send-stop-spotting-reports, 293
sensor, 1434
separation-distance, 572
separation-tolerance, 572
sessionId, 1674
setDeadReckoningAlgorithmToStaticOnPause, 1674
setMainThreadToHighPriority, 1675
showLines, 1139
simMgmtPduProcessLevel, 1675
sim-time, 354
simulation connection, 150
Simulation Model Sets, 231
Simulation Object, 762, 858
Simulation Terrain, 230
simulationModelSet, 1675
Simulation-Model-Set-Files, 354
simulation-run-duration, 272
siteId, 1675
slope-avoidance-factor, 508
slot, 1446
sms-filename, 271
soil-factors, 505
soil-list, 505
specifying for script, 758
specifying for task, 563
specifying simulation object, 861
speed-to-rpm-factor, 311
spheroidSemiMajor, 1675
spheroidSemiMinor, 1675
standard-power-type, 1560
standard-terrain-damage-type, 1560
startInRunMode, 1675
startMinimized, 1675
state-repository, 1684, 1686
statusUpdateHeartbeat, 1675
stopping-distance-factor, 1568
stopping-factor, 507
structure of entries, 1682
subnetMask, 1679
subsurface-entity-param, 1688
supplies, 1438
suppressDataAvailableException, 1675
suppression-degrades-sensor-performance-by, 522
suppression-detonation-insult, 522
suppression-detonation-range, 522
suppression-insult-level, 521
suppression-recovery-time, 521
suppressSelfReflect, 1679
surface-entity-param, 1688
tactical-smoke-param, 1477
targetFrameRate, 1676
targeting-control, 1528
task, Lua, 797
task-manager, 1685, 1687
taskVisualizationPublicationMode, 1676
terrain-check, 513
terrainDatabase, 1676
Terrain-Database, 353
tickPublishersDuringEntityTick, 1676
Time Multiplier, 232
time-multiplier, 353
tracer-type, 1529
types for script, 755
underFireDistance, 897

under-fire-parameter, 941
underFireTime, 897
useAbsoluteTimeStamps, 1676
useAdvisories, 1678
useAsyncIO, 1679
useCustomMapDatum, 1676
useDrawControlForTaskVisualizations, 1677
useIpv6, 1679
use-logger-control, 271
useSingleNetworkThreadWhenRtiThreadSafetyNotGuaranteed, 1678
using remote entities as, 945
vrf-base-object-param, 1501, 1682
wakeLength, 345
warhead, 1529, 1546
water-slide-depth, 507
water-slide-factor, 507
parameter property, 1490
adding, 1490
parameters
combat, 1437
parameter-type control object, 1501
parameter-type parameter, 1469, 1472, 1476, 1501, 1682-1683
paratrooper, 660, 684
Paratrooper Air Drop
task, 608
parent
disembarking from, 410
SMS, See simulation model set, included
particle effect
sea spray, 1107
passenger
embarking as, 683
slot, 683
passing-rounds-suppress parameter, 521
passive sonar, 310
pasting
appearance, 397
objects, 395-397
Paste Special, 397
performance and, 396
plans, 397
simulation object, 397
special, 397
altitude, 398
patch
soldier, 894, 908
terrain data, 1243
path
data, 1272
default Windows install, 97
disabling use of navigation data, 889
finding, 1274
following, 1274
MAK RTI, 107
planning, 500, 1274
nav mesh, 501
navigation preferences, 501
off road, 501
performance, 247
road, 501
using road data, 890
pathname
interpreting by object parameter database, 1502
scenario files, 355
Patrol Area task, 609
Patrol Between task, 609
Patrol Route task, 609
patrolling
area, 609
between objects, 609
route, 609
pattern
editing condition, 964
in condition, 963-964
loiter, flying, 592
Pattern Hold task, 610
pause mode, 1675
pausing
scenario
automatically, 232, 268
using PDU, 273
simulation, 262
PDB files
Linux, 98
Windows, 98
PDU
profiling, enable, 181
sending, 1172
siman, configuring, 1675
Start/Freeze, 273
Start/Resume, 273
Stop/Freeze, 273
Underwater Acoustics, 310
view control, 1208
pedestrian
area, 436
adding pedestrians, 652
creating, 434
path control object, 435, 994
pen style, 1033
Pen Style set data request, 1041
Percent Complete set data request, 923
performance, 207, 1192
affected by copy and paste, 396

alpha to coverage, 207
anti-aliasing, 164, 1660-1661
API support, 88
asynchronous I/O, 193
attached observer, 1214
clouds, 1660
coloring draw calls, 215
cube map calculation, 1096
developer debug overlay, 221
DI-Guy, 220
factors affecting, 196
fault tolerant heap, 1661
frame rate, 184, 215
function profiler, 202
graphics, depth, 164
HAT lines, 480, 1001
HBAO, 1660
HDR, 1660
HLA object discovery, 198
improving, 204
human entities, 881
load balancing, 197, 255
main thread, 183
multiple screens, 217
multiprocessor optimization, 197
network, tuning, 212
ocean, 1660
ocean planar reflections, 1093
optimizing, 196, 199, 1660
path planning, 247
preloading models, 218
setting application priority, 168
shadows, 1102
statistics
displaying, 202
VR-Forces, 202
stencil bits, 174
tactical graphics, 1000
terrain, 165, 221, 223, 1660
Tracy Client, 203
visual, 216
visual for 3D application, tuning, 212
zooming in, 1193
Performance Monitor Panel, 200
performance settings, 213
period, burst, 421
periodic
checkpointing
disabling, 266
scheduling, 266
snapshot, enabling, 267
periscope
lowering, 702
raising, 702
permanent intervisibility object, 1124
persistent shadow technique, 166
Personnel and Supplies set data request, 923
personnel parameter, 1438
personnel, status, 465
perspective in display, 1609
perspective view, model, 1353
phase line, 994, 996
Entity Left of Line, 940
phase, synchronization matrix, 987
physical model, 367
physical parameter, 1431
physicalWorldParams.mtl
file, 1513
Pick Up Passengers at Location task, 669
pinning
range rings, 1068
sensor contact lines, 1064
task visualization, 1062
track history
spot report, 297
tactical data link message, 300
pitch, 1177
setting, 892
manually, 389
viewing, 389
Pitch set data request, 892
Place IED task, 643
placing
IED, 643
new entity, 386
props, 382
simulation objects, 382
tactical graphics, 382
plan
abandoning, 554, 982
confirmation prompt, 561
from a plan, 982
altitude condition, 939
assigning to multiple simulation objects, 980
concepts, 930
condition
If, 950
Trigger, 951
Wait Until, 951
While, 950
conditional statement, 933
considerations for creating, 948
console message, sending, 972
coordinated, 984
copying, 981
creating object, 973

custom, 416, 421, 425
default, 416, 421, 425
deleting
object, 973
simulation object, 973
statement, 955
Do Until Interrupt condition, 938
drag and drop statement, 981
editing, 948, 982
statement in, 955
entity, 932
file, 225, 350
editing, 356
Follow Entity task, 945
force hostility, 302
global, 932
writing, 966
global command, 969
global task, view control, 975
If condition, 935
including scenario events, 284
inverting condition parameter, 963
issuing, 973
manager, 1685
Move To task, 604
name in condition, 964
pasting, 397
pattern in condition, 964
printing, 955
remote simulation object, 945, 980
restarting, 981
saving, 955
simulation time condition, 942
statement, 949
task, 949
Tasked by Superior, 945
test result, 181, 979
trigger, 935, 951
unit, for, 980
variable, 943
viewing, 943
Wait Until condition, 938
When statement, 935
While condition, 937
writing, 948
Plan parameter, 353
Plan View, 137
north, 1199
observer mode, 1164-1165
orient north, 1199
plan view, 1171
planar reflection, 1092
SpeedTree, 1093
plan-manager parameter, 1685, 1687
platform
file, 1481
variable binding, 1486-1488
specifying, 1342
update, applying, 1326
platoon
dismount, 670, 909
marking, 1357
mount, 671, 910
preconfigured, 440
simulation object icon, 1080
playing a scenario event, 283
plug-in
adding, 158
configuration
adding, 160
deleting, 161
DLL
adding, 159
removing, 160
specifying, 159
initializer, 167
list, viewing, 162
loading, 157, 166, 170, 173, 180-181, 190, 1306
managing, 156
version
checking, 166
skip check, 166
XML file, 158
deleting, 161
pluginsDirectory parameter, 1673
PNG, 1243
point, 996
civilian visit, 436
crowd gathering at, 653
feature, 1244
feature list, 1258
on surface entity, 996
orbiting around, 1186
patrolling between, 609
reversing, 1033
riot, 662
sink, 418
spawn, 417
strong, constructing, 725
suppressive fire at, 520, 637, 645
terrain profile
sampling rate, 1267
point-to-point communications
configuring, 1678
specifying, 192
point-to-point intervisibility, 1123

POL status, 465
polarized stereo, 1616
  configuring, 1617
policy
  Clamp Near Clip When Magnified, 1606
  Distance To Attached, 1607
  Reduce Near Clip, 1606
  Reduce Z Fighting, 1606
polygon, 997, 1246
  not going through, 1200
port, 366, 1478
  default, 192
  for scenario files, 1669
  group, 366, 1478
  number, 1679
    specifying at startup, 192
  Tracy Client, 184
position, 1177
  camera, 1615
  observer, 1187, 1191, 1214
  window attribute, 1603
position-offset parameter, 1566
post processor settings, 1097
posture
  adding, 1440
  affect on footprint, sensing, sensor signature, speed, 530
  changing, 733, 924
  Deliberate-Attack, 530
  Deliberate-Defense, 530
  embarked entity, 1447
  Hasty-Attack, 530
  Hasty-Defense, 530
  Reconnaissance, 530
  Rout, 530
  specifying, 892
  transition time
    editing, 1434
    urgency, 908
  Travel, 530
  unit, 530
Posture set data request, 892, 924
power
  adding new, 1553
  combat, 528
  determinant, 1547
  explosive, 1535
  laser, 886
power-list block, 1547
power-plant-active parameter, 310
power-type parameter, 1561
praying, human task, 650
precedence of components in OPD, 1481
precipitation, 324, 333
  intensity, 328
  rain on terrain, 329
  snow on ground, 329
  type, 328
preconfigured unit, 440
Prefer Roads, 890
preferred movement mode, 664
preferred road data, 664
Preferred Trajectory Height set data request, 893
prefix notation, 956
preloading models, 218
Prepare QRA task, 731
previewing damage, part articulation, lights, and switch node states, 1353
primary simulation object, 1216, 1219
printing
  plan, 955
  window, 138
priority
  application, 168
  simulation model sets, 1312
  weather zone, 336
prisoner-of-war, 903
probability
  strike target, 1539
  table, direct and collateral damage, 1531
probability-coefficient list, 1530
probe-and-drogue refueling, 701
process
  background, 812
  ID, front-end, 181
  object mappings to, 357
  VR-Forces, 114
product installation, 96
  build information, 96
  Linux, 97
  Windows, 96
proficiency, weapon systems, 900
profile, navigation data, 1281
profiler function, 202
projected terrain database, 1182
projection
  2D or 3D, 1164
  supported, 85
Projection Resize Policy, 1604
  configuring, 1607
Projection Units, 1604
promote
  aerodrome, 1293
  runways, 1293
Promote From Clutter set data request, 893
promotion-id parameter, 1566

prompt, confirmation
abandon plan, 561
quit, 146
task, enabling and disabling, 561
prone, human posture, 892
prop, 382, 1244
adding
to Props Palette, 402
attaching observer to, 1210, 1214
configuring, 1244
creating, 372-374, 377, 381-382, 402
extracting
from terrain, 1244
filtering attachment list, 1213
heading, specifying, 389
pitch, specifying, 389
placing, 382
saving, 402
property
adding, 1490
buttonGroup, 857
channel, 1604
editing system, 1392
object, specifying, 372
parameter, 1490
scenario, 251
state, 1490
user-defined, 1490
tactical graphic, default, 1364
window, 1603
Props Palette, 372-374
adding objects to, 402
propulsion noise, 310
Protest Along Line task, 661
Protest Around Location task, 662
protesting
crowd, 661-662
human, 651
Provide Indirect Fire task, 644
Provide Mobile Overwatch task, 674
Provide Stationary Overwatch task, 675
Provide Suppressive Fire from Cover task, 520
Provide Suppressive Fire on Location task, 520
Provide Suppressive Fire task, 645
proximity detonation, 1536
proximity fuse, 875
proximity sensor, 1536
pruning
element definitions, 1359
model definitions, 1359
object type mappings, 1359
visual models, 1359
publish time, back-end statistics, 200
published object type, 1470
publisher
entity ticks, 1676
parallel or synchronous, 1670
publishing
entity velocity when paused, 1673
radio trasmitter, 1574
tactical graphics, 1019
publish-transmitter parameter, 1575
publish-transmitters parameter, 1574
publishZeroVelocityWhenPaused parameter, 1673
puddles, rain, 329
pulse repetition rate, 1136

Q

QRA, See quick reaction alert
Qt Designer, 854, 1488
design elements, adding, 859
parameter, adding, 854
selecting for script dialog box use, 854
UI file details, 1489
Qt Linguist, 108
quadcopter, 611-612
quality
ocean, 346
question
asking from script, 805
scripted task, 485
queue
releasing entity from, 706
waiting in, 664
quick reaction alert
launching aircraft, 731
preparing aircraft for, 731
quiet mode, 180
quit prompt, disabling, 146
quitting
plan, 982
retask procedure, 563

R

radar, 894
affect of rain, 1516
coverage, 306
configuring, 309
ID, 878
jamming, 306, 539-540
sensor, 878
radar coverage
displaying, 308

enabling, 308
radar mode
search, 894
track, 894
Radar Mode set data request, 894
radar sensor, sector blanking, 897
Radar-Jamming-Strength-Receiving, 540
radio, 870, 875, 1148
communication line, 1073
color, 1076
enabling and disabling, 1074
lifetime, 1076
configuring, 1574
message
set data request, 695
task, 696
Radio Button Choice parameter, 762, 857
radio transmitter, publishing, 1574
radioMessageTypes.h, 1645
radius
emitter volume, 1137
sheaf, 718
rain, 324, 333
accumulation, 329
effect on radar, 1516
intensity, 328
puddles, 329
splash effect, 1106
weather zone, creating, 337
rain-to-attenuation-coefficient parameter, 1516
Raise Periscope task, 702
random
DI-Guy appearance, 1385
simulation object, 400
Random condition, 941
Random Number Seed parameter, 232
randomized simulation object, 1396
random-number-seed parameter, 353
range
bullseye, 1026
focal, 1233
ground clamping, 1249
laser codes, 304
measurement unit, 1634
range line, 1130
configuring, 1132
creating, 1130
deleting, 1132
range ring, 1065
bullseye, 1028
configuring display of, 1068
displaying, 1065
displaying for entity, 1068
enabling and disabling, 1065, 1067
pinning, 1068
sensor, 1065
spider web, 1015
range-coefficients determinant, 1549
range-determinant parameter, 1530
range-from parameter, 1547
range-list determinant, 1549
range-list parameter, 1530
Rank Patch set data request, 894
rate
adaptive tick, 217
climb, turn, and descent, 591
turn, 592
ray
crepuscular, 1095
intervisibility, 1123
sun, 1095
RCI, See remote control
reactive task, 436, 572, 707, 746, 789, 811
canceling, 576
creating, 772
disabling, 574
enabling, 574
managing, 576
Visit Interest Points, 436
reader writer key value pair, 1440
Real DB, 92
real time
checkpointing in, 266
running faster or slower than, 123
rearview mirror, 1612
recalling
view, 1204
view state, 1202
receive buffer size, 177, 182, 1150, 1679
Receive Text Message condition, 942
receiveInfluence() function, 789
receiveObjectMessage() function, 788
receiveTextMessage() function, 789
receiving
radio messages, 1574
supplies, 538
recently used scenario, 251
Reconnaissance posture, 530
recovering embedded entity, 703
recovery, 873
recvBufferSize parameter, 1679
redo ellipse rotation, 1032
Reduce Near Clip policy, 1606
Reduce Z Fighting policy, 1606
Reduce Z Fighting Ratio attribute, 1606

reference ellipsoid, 1669, 1675
  configuring, 1676
reflection
  map, 86
  planar, 1092
refresh rate, 216
refreshing
  visual models, 1360
refuel
  aircraft, 587, 610
  supply, 1391
  tanker aircraft, 692
Refuel from Tanker task, 738
Refuel From task, 610
refueling
  aircraft, 738
  boom, 700
  drogue, 701
Refueling Mission task, 739
regenerateNavigationDataAtRuntime parameter, 1673
regenerating navigation data, 1284
regenerating spot report definition, 1349
registering trigger, 937, 953
regulating time, 142
reinitializing a scenario, 264
rejectMismatchedScenarioMessages parameter, 1673
rejoining exercise, 118
relative path, 355
relative-heading parameter, 412
relative-position parameter, 412
Release Bomb on Laser Spot task, 646
release bomb on location, 646
Release Bomb on Location task, 646
release bomb on target, 646
Release Bomb task, 646
Release Next Entity From Queue task, 706
Release Unguided Bomb task, 647
reloading
  visual models, 1360
remapping
  back-ends, 119
  batch mode, 272
  simuilation objects, 248
remapping objects, 258
remote
  attachment, key-names, 1593
  display on multiple screens, 217
  entity, attaching components to, 230, 1592
  object, 120, 1473, 1685
    network interface, 1477
  observers, 1172
  simulation object, 120, 558
    markings, 945
    plan, 980
    use in plans, 945
  stealth, displaying models, 1173
remote control
  API, 88
    MAK Data Logger, 273
    mission management, 543
  interface, 87
  of the observer, 1207
  view control, 975
remote discovery, disable, 175, 180
remote graphics, keyword, 1613
Remote Sensor Receiver system, 895
Remote Sensor set data request, 895
remoteAttachment example scenario, 1592
remote-attachment-scheme parameter, 354
remote-attachment-scheme-data parameter, 354
remoteConfigurations directory, 1593
remote-configurations entry, 1593
remote-objects parameter, 1473, 1685
Remove Jamming Target set data request, 895
removing
  articulated parts, 1389
  assembly, 1428
  channel, 1602
  force, 1333
  formation, 1413
  navigation area, 1280
  object from OOB, 494
  objects from selection group, 460
  observer, 1167, 1170
  simulation objectfrom unit, 441
  subordinate, 1409
  system, 1392
  tag, 1327, 1329
  targets from priority list, 906
  triggers from plan, 954
  view, 1204
  window, 1602
renaming
  formation, 1413
  observer mode, 1167
  scenario, 249
  selection group, 460
  tag, 1327, 1329
render bin, 208
Reorganize set data request, 895
reorganizing
  concepts for units, 516
  OOB members, 495

unit, 895
versus closing formation, 517
repair, 873
replacing
saved view list, 1206
subordinate, 1409
system on simulation object, 1393
replicating settings and data, 126
report
NBC, 739
obstacle, 739
reportedHostAddressString parameter, 1674
report-resend-period parameter, 294
repository, 1684
require statement, Lua, 787
resetting
simulation object health and stores, 896
resizing
boxes and text overlay objects, 1032
windows, 1598, 1607
resource
displaying, 481
information, 463
monitoring entity, 481
pacing and tracking, 925
specifying, 896, 920
status at creation, 391
Resource condition, 942
resource-item, 1440
resources
measurement unit, 1634
Resources Pacing/Tracking set data request, 925
Resources set data request, 896
Restart Plan command, 981
Restore set data request, 896
restoring
embedded entity parent, 414
simulation object health and stores, 896
unit, 896
view, 1204
result
detonation, 1562
resume() function, 788
resuming a simulation, 260
resupply, 539, 873
modes, 925
Resupply Mode set data request, 925
Retract Towed Array Decoy task, 702
retreat, 734-735
retrograde movement, 734-735
Return to Base task, 611
reuseEntityIdentifiersOnScenarioLoad parameter, 1674
reverse Z buffer, 1611
disable, 165
reversing
line direction, 1033
movement along route, 598, 600, 603
reverting
category changes, 1329
navigation area changes, 1280
simulation object changes, 1345
window layout, 1627
revision, RPR FOM, 151, 176, 186, 1150
revolving the observer, 1186
rewinding
scenario, 262, 264
scripted task, 775
to a snapshot or checkpoint, 262
RGB, 1243
RID file, 107, 190
rid.mtl file, 107, 190
right-click menu, 1630
configuring, 1643
rightclickmenu element, 1643
ring, range, 1065
configuration, 1068
enabling and disabling, 1067
riot point, 662
Riot task, 662
road
data, use in path planning, 890
driving on, 503
left or right side, 504
network, 503
using route as, 599
Role set data request, 926
roll, 1177
roll up assemblies rule, 1428
rolling back to a snapshot, 262
rolling up assembly, 1428
rope
extraction, 686-687
insertion, 688
sliding down, 686
sling load drop off, 685
sling load pick up, 685
Rope Extraction task, 686
Rope Sling Load Drop Off task, 685
Rope Sling Load Pick Up task, 685
Rope Suspention Insertion task, 688
Rotary Wing Airlift Cargo to Location task, 611
Rotary Wing Land task, 612
Rotary Wing Retrieve Cargo task, 612
Rotary Wing Unload Cargo task, 612

rotary-wing entity
  altitude, 589, 591
  attacking with gun, 625
  circling a point, 607
  heading, 591-592
  holding pattern, 610
  landing, 612
  loiter pattern, 589-590, 592
  measurement unit, 1634
  rotor wash, high quality, 1110
  speed, 464, 473
  terrain avoidance, 514
rotary-wing-entity-param parameter, 1476, 1688
rotary-wing-entity-state-repository, 1476
rotating an ellipse, 1032
rotation path, smoothing, 213
rotation, wheels, 1078
rotor wash, 1106
  enabling and disabling, 1109
  high quality, 1110
roughness
  mapped to soil type, 505
round, 314
round-down parameter, 1552
Rout posture, 530
route, 994, 997
  altitude, 1000
    displaying, 1001
    setting, 388
  computed, 1019
  extruded, 1016
  following, 599-600
  patrolling, 609
  terrain profile, 1267
  using as road, 599
RPR FOM, 191
  revision, 151, 176, 186, 1150
  specifying version, 1678
  version, 151, 176, 186, 1150, 1678
rprFomVersion parameter, 1678
RTB, See Return to Base task
RTI, 106
  configure for time management, 143
  prevent check, 186
  thread safety, 1670, 1678
RTI_CONFIG environment variable, 107
RTI_timeMgmt parameter, 143
RTI_useRtiExec parameter, 143
rtiexec, 143
rule for rolling up assemblies, 1428
rules of engagement
  pasting, 397
  specifying, 896
Rules of Engagement set data request, 896
run mode, 1675
  sim. engine, 183
run-duration parameter, 272
run-duration-time parameter, 354
runInTimeManagementMode parameter, 144, 1678
running, human posture, 892
runway, 512
  active, 1292
  conditions, 1298
  damaged, 1298
  lights
    set data request, 1298
  promote, 1293
Runway Active set data request, 1297
Runway Arrestor Gear set data request, 1297
Runway Condition set data request, 1298
Runway Damaged set data request, 1298
Runway Light Conditions set data request, 1298
Rush to Location task, 662
rwKeywordValuePairs.xml, 1440

S

Sadr City terrain, 1239
Sail Heading task, 568, 613
salvo, 314
sampling rate
  freehand line, 1012
  intervisibility, configuring, 1127
sandstorm, 337
satellite, 246
  setting orbit, 890
saturation-water-depth parameter, 507
saved views file, 225
  scenario, 350
saveState() function, 788
saving
  checkpoint, 265
  display engine configuration, 1602
  global variables, 794
  load-balanced scenario, 248
  observer view, individual, 1206
  plan, 955
  prop, 402
  scenario, 118, 247, 264
    existing, 247-248
    new name, 249
  screen capture, 138
  scripts, 775
  settings, 123, 125

tactical graphics, 1007
view, 1202
to file, 1206
window layout, 1624, 1627
scale
2D icon, 1080
3D model, 1082
factor, 1082
material ambient and diffuse value, 1089
model, linear and XR, 1082
scatterable mines, 1021
scenario, 225
batch, 270
building blocks, 238
checkpoint, 264-265
closing, 263
collaborative development, 239
component attachment table, 232
compressed format, 225, 350
creating new, 228
creating objects from OOB, 493
description, editing, 253
directory, 254
directory, user, 174, 254
end time, 232, 268-269
event, See scenario event
extras file, 422
file, 119, 225, 350
editing, 351
parameters, 351
pathnames, 355
specifying which to use, 1674
format, 249
frame mode parameter, 232
frame time parameter, 233
GUI settings file, 350
importing, 238-239
information, displaying, 252
initial information, 236
load balancing
delete simulation object, 232
existing, 255
manually, 257
new, 255
ownership transfer, 257
rule set, 256
load-balanced
saving, 248
starting, 258
loading, 173, 182, 249-251
at startup, 182, 251
recently used, 251
merging, 239
MSDL
exporting, 241
importing, 241
name, 230
objects, See scenario objects
parameter
setting, 230
version, 355
path planning, 247
properties, editing, 251
random number seed, 232
reinitializing, 264
remoteAttachment, 1592
renaming, 249
reopening, 264
resuming, 260
rewinding, 262, 264
scripted task, 775
rolling back to snapshot or checkpoint, 262
running, 260
saving, 118, 247, 264
existing, 247-248
new name, 249
scenario objects, 357
simulation model set, 231
multiple, 233
snapshot, 264
spawn pattern, 422
starting, 260
starting date and time, 236
startup observer view, 1205
temporary directory, 358
terrain, 356
display, 231
simulation, 230
time multiplier, 232
zoom to extents on load, 1195
scenario event, 276
adding audio, video, and graphics, 279
creating, 276
deleting, 287
enabling and disabling, 286
ending, 285
intelligence object, 281
sending, 283
sending from plan, 284
showing and hiding, 287
Scenario Event condition, 942
scenario extras file, 350
Scenario Name parameter, 230
scenario objects, 357
file, 350, 357, 1007
default values in, 358

scenario script, 745
Scenario Startup dialog box, enabling and disabling, 132
scenario-data parameter, 355
ScenarioDescription parameter, 355
ScenarioExtentInformation parameter, 355
Scenario-Extras parameter, 353
scenarioFileName parameter, 1674
scenario-filename parameter, 271
scenario-name parameter, 354
Scenario-Scripts parameter, 353
scheduling
  crowd gathering, 653
  periodic checkpoint, 266
  spawn event, 419
schema
  tactical graphic model, 1366
scnx file, 225, 350
screen
  capturing image, 138
  multiple, performance impact, 217
  number, specifying, 170
screen capture
  Create Object Palette, 1362
  saving, 138
  setting save location, 138
Screen Number attribute, 1603
script
  adding to toolbar, 769
  availability by simulation object type or behavior set, 766
  creating, 747
  deleting, 779
  description, 754
  editing, 782
  execution, 787
  extended name, 754
  ID, 768
    editing, 753
  image list, 762
  laying out using Designer, 854
  loading, 787
  location, 801
  making available by behavior set, 767
  metadata, 747
  parameter types, 755
  parameter, specifying, 758
  previewing dialog box, 749
  saving, 775
  script ID, 750, 753
  set data request, 745
  set, toolbar location, 769
  simulation object type supported, 767
  system and scenario, 745
  system, directory, 178
  task
    menu location, 764
    toolbar location, 769
  text editor, specifying, 781
script editor, 746
scripted
  behavior, 585
  movement
    configuring, 1399
scripted set, 746
  accessing state properties, 1493
  creating, 771
scripted task, 745-746
  accessing state properties, 1493
  actioncategory, 581
  adding to folder, 777
  adding to toolbar, 1646
  background process, 812
  concurrent, 771
  copying, 779
  editing, 776
  entry point, 787
  execution flow, 790
  exporting and importing, 777
  filtering list, 776
  folder
    adding, 777
    deleting, 777
    renaming, 777
  importing, 778
  in system definition, 768
  interactive, 805
  language, 770
  organizing, 777
  package, 777
  parameters, list of, 800
  question, 485
  reactive task, 572, 789, 811
  removing from folder, 777
  rewinding, 775
  script ID, 796
  script storage location, 777
  system
    creating, 779
    making available on Task menu, 780
    storage location, 779
  Update_EW_Degradation, 540
script-ids parameter, 768
scripts file, 350
SDK toolkit, 99
  documentation, 97

sea
  choppiness, 333
  spray, 1106-1108
    enabling and disabling, 1107
  spray, droplet size, 1108
  state, 333, 344
    Douglas, 341
  swell, 333, 344
search
  radar mode, 894
  sector, 703
search and rescue, 703
  emergency beacon, 878, 1391
  flight pattern
    creeping line, 613
    expanding square, 615
    parallel, 616
    sector search, 618
  sensor, 1391
Search and Rescue task
  Fly Creeping Line Pattern, 613
  Fly Expanding Square Pattern, 615
  Fly Parallel Pattern, 616
  Sector Search, 618
secondary
  simulation object, 1216, 1219
  window, 137
section (unit)
  simulation object icon, 1080
sector
  area, 1014
  navigation area, 1277
  responsibility, 898
  spider web, 1015
  unit, 529, 1065
Sector Blanking set data request, 897
Sector of Responsibility set data request, 898
Sector Search Operation task, 703
Sector Search pattern
  Search and Rescue task, 618
segment
  angle, specifying, 1138
  arc size, 1138
  edge, 1139
  emitter volume, 1137
  outline, displaying, 1139
Seize Objective task, 678
Selected Simulation Engine Toolbar, 375, 377
selecting
  window layout, 1625
selection box, 455
selection group
  adding object to new, 459
  adding to, 460
  creating, 458
  creating from keyboard, 459
  creating from menu, 459
  deleting, 460
  file for scenario, 350
  introduction, 457
  removing object from, 460
  renaming, 460
  selecting, 459
Selection Group View, 452
SelectionGroups parameter, 353
self condition parameter, 963
self reflection, DIS, 187, 1150
send buffer size, 177, 182, 1150, 1679
Send Radio Set task, 695
Send Radio Task task, 696
Send Text Message command, 972
Send Text Message task, 697, 972
sendBackendLogToNetwork parameter, 1674
sendBufferSize parameter, 1679
sendFedTime parameter, 1678
sending
  console message, 972
  data, configuring, 1670
  messages, 367
  NBC report, 739
  obstacle report, 739
  radio messages, 1574
  scenario events, 283
    from plan, 284
  set data request via radio message, 695
  task via radio message, 696
  text message (command), 972
  text message (task), 697, 972
  view control in a plan, 975
sendMessagesInNetworkProcess parameter, 1674
send-only-to-front-end parameter, 293
send-spot-reports-on-own-force parameter, 293
send-stop-spotting-reports parameter, 293
SenSim, 1116
sensing
  electronic emission, 539
  target, 1528
sensor, 302, 366, 1504, 1528, See also sensor signature
  affect of posture, 530
  aiming, 899
  apparent signature, 1510
  attaching to remote entity, 1592
  channel attribute, 1604
  component, 1514
  configuring, 1504

connecting, 1504
detection probability, 1510
detection table, 1516
detection types, 1512
emission, 541
footprint, 533
Full Color, 1119
gimbaled, 899, 1228
field of view, 1229
field of view color, 899
setting azimuth, 1232
information, 463
new, 1513
parameters, 1434
proximity, 1536
radar, 878
sector blanking, 897
range ring, 1065
sensor contact line, 1063
SIGINT, 541
signature, apparent, 1516
signature-object-sensor, 1504
system, 1390
system definition file, 1482
editing, 1505
unit, 533
view, 1228, 1230
create command, 978
visual, illumination, 1515
zooming, 899
Sensor Aim set data request, 899
sensor component, adding, 1514
sensor contact line
configuring, 1065
enabling, 1063
enabling and disabling, 1063
pinning, 1064
specifying objects that can display, 1064
Sensor Control Panel, 1230
sensor effects, 1116
adding, 1119
blur, noise, and gain, 1117
configuring, 1117
Sensor Enabled set data request, 902
Sensor Field of View Color set data request, 899
sensor signature, 310, 1391, 1506
adding, 1515
affect of posture, 530
modifying, 1020
predefined, 1387
setting, 1387
standard, 1507
Sensor Texture map, 1117
sensor view, 1228
create command, 978
Sensor Zoom set data request, 899
sensor-signatures.csv, 1387, 1507
separation-distance parameter, 572
separation-tolerance parameter, 572
separator, menu, 1640
server
terrain, 1246, 1248
session, 115
ID, 115, 129, 133, 150, 152, 176-177, 183
assigning, 133
configuring, 1674
setting at startup, 183
joining at startup, 140
message, configuring, 140
starting and stopping, 273
sessionId parameter, 1674
set
scripted, 746
creating, 771
Set command
default, 861
set data request, 861
Activation, 1038
Active Sonar Mode, 866
Add Jamming Target, 866
Aerodrome Light Conditions, 1295
Aerodrome Undercarriage Inspection System, 1295
AI Enabled, 867
Aiming, 868
Aircraft Lighting, 868
Allow Crossing set, 1039
Altitude, 510, 869
Ammunition, 915
Ammunition Pacing/Tracking, 915
Appearance, 869, 1090
Armed, 870
Arrowhead Style, 1039
Articulated Part, 870
Assign Aircraft, 916
Assign Transport, 909
Assume Unknown is Hostile, 870
Attached, 871
Attention Grabbing, 871
Automatically Close Doors, 872
Autonomous Actions enabled, 866
Base, 872
Callsign, 873
Capabilities, 873
Collision Avoidance Enabled, 873
Color, 1040
Concealed, 873

Contamination, 918
Countermeasures Auto Launch, 311, 874
Current Speed, 874
Damage Percentage, 875
Destroyed, 875
Detonation Fuse Type, 875
DI-Guy Characteristics, 876
DI-Guy Hand Item, 876
Disembarked, 410, 877
Dismounted, 909
Dismounted Headquarters Section, 909
Embarked, 407, 877
Emergency Beacon, 878
Emitter, 737, 878
Engagement Results, 919
Engagement Zone, 879
Equipment Pacing/Tracking, 919
Facilities, 919
Fill Style, 1040
Food, Water, Fuel, Oil, and Lubricant Pacing/Tracking, 921
Food, Water, Motor-Gas, Aviation-Fuel, Diesel Fuel, Oil, Lubricant, 920
for IFF transponder, 883
Force, 879, 1040
Formation, 879
  in plan, 945
Freeze Fuel Levels, 880
Freeze Movement, 880
Freeze Munition Levels, 881
GPS Jammer, 881
Ground Clamp Fidelity, 881
Has Drag Chute, 921
Heading, 882
Hide Occupants, 882
Invisible, 884
Invulnerable, 884
Jammer State, 885
Jamming Mode, 885
Lase Autonomous, 305
Laser Code, 304-305, 885
Laser Power and Wavelength, 886
Light (Advanced), 1295
Line Width, 1040
Loadout, 886, 922
Location, 887
Lock Embarkation, 910
Look At, 888
MOPP Level, 922
Morale, 923
Mounted Crew, 910
Mounted Headquarters Section, 910
Nametag, 889
Navigation Enabled, 889
Navigation Preferences, 890
Notify Level, 890
  tactical graphic, 1041
one click, 571, 751
Orbit, 890
Ordered Formation, 911
Ordered Speed, 568, 891
Pen Style, 1041
Percent Complete, 923
Personnel and Supplies, 923
pinning, 564
Pitch, 892
Posture, 892, 924
Preferred Trajectory Height, 893
Promote From Clutter, 893
Radar mode, 894
Rank Patch, 894
Remote Sensor, 895
Remove Jamming Target, 895
Reorganize, 895
Resources, 896
Resources Pacing/Tracking, 925
Restore, 896
Resupply Mode, 925
Role, 926
Rules of Engagement, 896
Runway Active, 1297
Runway Arrestor Gear, 1297
Runway Condition, 1298
Runway Damaged, 1298
Runway Light Conditions, 1298
script, 745
Sector Blanking, 897
Sector of Responsibility, 898
sending by radio, 695
Sensor Aim, 899
Sensor Enabled set, 902
Sensor Field of View Color, 899
Sensor Zoom, 899
Skill Level, 900
Sonar Depth, 900
Spot Reports, 901
Stow Hand Item, 902
Subsystem Enabled, 902
Superior, 902
Supplying, 926
Surface Vessel Lighting, 903
Surrendered, 903
Synchronize Laser Code, 305, 903
TACAN transmitter, 904
tactical graphics, 1038
Tanker Lighting, 904

Target, 905
Target Priority List - Add, 905
Target Priority List - Clear, 906
Target Priority List - Update, 906
Tasked by Superior, 572, 906
TDL Messaging, 907
toolbar, adding to, 1646
Tracer Use, 907
Undetectable, 907
Unit Patches, 908
Unloaded, 911
Urgency, 908
Weapons Pacing/Tracking, 927
Set menu
editing, 1645
icon, 753
sorting, 1640
set statement, See set data request
setCheckpointMode function, 794
setDeadReckoningAlgorithmToStaticOnPause parameter, 1674
setMainThreadToHighPriority parameter, 1675
settings
3D, 217
air pressure, 327
air temperature, 327
altitude, 386-388
date, 331
environment configuration, 167
factory and default, 123
GUI layout, 168
GUI, saving, 1624
GUI, SOE, 1307
IFF, 883
loading, 123
local, RTI, 176, 186, 1151
location of saved, 125
LOD scale, 1093
notification level, 189, 890
entity, 483
post processor, 1097
precipitation
intensity, 328
type, 328
property, default, 1364
saving, 123
scenario GUI, 350
segment arc size, 1138
shared directory, 173, 183, 1308
Simulation Object Editor, GUI layout, 1307
simulation speed, time management, 260
synchronizing, 126
time of day, 332
time zone, 332
unit bounding volume color, 450
user, 125
directory, 174, 185, 188, 1150, 1308
visibility, 327
weather, 324
SFOV, See sensor view
shader, 86
shadow
configuring, 1102
enabling and disabling, 1100
label text, 476
persistent, enable, 166
SpeedTree trees, 1102
shaking camera, 1198
shape terrain data, 1243
shapefile
exporting overlays to, 1008
importing to overlay, 1009
shared data
directory, 173
root, 173
stem, 173
shared resources, 204
shared settings directory, 173, 183, 1308
sheaf radius, 718
ship, See also watercraft
waypoint, 996
Ship Clutter Area, 429
shooter, laser, 304
shooter-and-impact-point range, 1548
shooter-to-impact range, 1548
short name, 1340
shortcut menu, 1630, 1643
Show Function Profiler Overlay, 202
Show Occupants, 683
Show Session Terrain Change Prompts, 141
showCockpits keyword, 1613
showHuds, 1604
showLines parameter, 1139
showRemoteGraphics keyword, 1613
SHP file, 1243
shutdown() function, 788
SIDC
map to DIS, 182
SIGINT sensor, 541
signal
interaction, 1073
signal_slot, 1489
signature
apparent, 1510, 1516
sensor, 310, 1506
signature propagator, configuring, 1513

signature-object-sensor, 1504
SilverLining, 90
  configuration, 90
  license, 90
sim engine
  enabling time management, 144, 186, 192
simMgmtPduProcessLevel parameter, 1675
SimObject class, 784
simple-radio-comm-model, 365
Simthetiq, 92
sim-time parameter, 354
simulated munition, 1393
simulated world, navigating, 1177
simulation
  closing, 263
  connection
    parameters, 150
  date, 236
  edit plan, 982
  engine, 114
  object, See simulation object
  pausing, 262
  resuming, 260
  rewinding, 262
  running, 260
    automatically, 269
  speed with time management, 260
  start mode, 1675
  starting, 260
  time, 120, 142, 236
    checkpointing in, 266
    evaluating in plan, 942
Simulation Connections Configuration, 150, 187-188
Simulation Control Toolbar, 135
Simulation Date/Time condition, 942
simulation effect object, creating, 374
simulation engine, See also back-end
  closing, 146
  console commands, 1660
  exiting, 146
  remapping objects to, 258
  specifying, 375, 377
simulation management message, 1675
simulation model set, 1309
  aggregate level, 528
  child, See simulation model set, derived
  configuration, 235
    default, 235-236
  creating, 1323-1324
  default, 164
  deleting entity, 1362
  derived, 1310
  entity-level and aggregate-level, 368
  file, 1309
  hostility file, 301
  included, 1310
  Lua scripts, 1313
  migrating, 1321
  multiple, 233
  object list, 1326
  opening, 1323
  parent, See simulation model set, included
  platform file, 1481
  priority, 1312
  promote object from included, 1314
  promote system from included, 1316
  read-only, 1311
  specifying on command-line, 183
  updating, 1321
  vrfSim.opd, 1314
Simulation Model Sets parameter, 231
simulation object, See also entity; unit
  adding to
    order of battle, 491, 493
    selection group, 460
    unit, 440
  articulated parts, 1387
  attaching tactical graphic to, 1028
    automatically, 1029
  attachment list, filtering, 1213
  attribute, information, 463
  behavior, 367
  centering observer on, 1197
  changes, undoing, 1345
  communication model, 365
  components, 1477
  concepts, 362
  console message, 483
    notification level, 890
    saving, 485
    sending, 972
  copying, 395-396
  count, 487
  creating, 372-373, 377, 382
    from existing, 1361
    from OOB, 493
    in plan, 973
    multiple, 432
  creation, enabling and disabling, 1341
  damaged, 875
  deleting, 399
    from plan, 973
  destroyed, 875
  dragging, 392
    to new location, 887

editing, 393, 1305
enumeration
published and matching, 1470
filtering list of, 565
frame rate, 200
heading, 882
setting, 389
hiding, 290
in Simulation Object Editor, 1314
hostility, 300
icon, 1080
CID level and color, 303
identifiers, 362
information, 462
multiple, 468
inset view, 1224
intervisibility fan, 1124
label, 364, 473
extended, 477
list of, 1326
hiding in Simulation Object Editor, 1314
local and remote, 120
model, 1309
editing, 1337
reloading, 1360
move to entity, 603
move to location, 604
name, 469, 1340
length restrictions, 363
observer, attaching, 1214
pasting, 397
pinning intervisibility object to, 1128
pitch, specifying, 389, 892
placing, 382
plan
issuing to, 973
issuing to multiple, 980
viewing, 943
platform, updating, 1326
property
editing, 1383
keyword, 1383
specifying at creation, 384
random creation of, 400
range ring, 1065
reinitializing, 399
remote
in task, 558
plans, 980
rename, 1340
resource at creation, 391
resource, monitoring, 481
restoring health and stores, 896
selecting, 453, 455
from Object Console Summary Panel, 456
in Objects List Panel, 456
setting
appearance, 869
location, 887
state and parameters, 861
setting icon for icon list, 1344
specifying resource, 896
spot report, 296, 901
state, 367, 861
storing list of, 357
systems
adding, 1391
editing, 1390
replacing, 1393
tag, changing, 1337
target detection, 302
task, 367
tasking by superior, 906
track history, 1054
trailing effect, 1051
undoing changes, 1345
unit, removing from, 441
unselecting, 457
UUID, 362-363
viewing in Navigation Lab, 1289
views of, 452
waiting, 698
ways to add to scenario, 238
writing plan, 948
zooming to extents, 1194
simulation object creation palette, 373-374
Simulation Object Editor, 89, 1303, 1305
command-line arguments, 1306
DI-Guy character and appearance, 1383
embedded entity, 411
force
adding, 1331
editing, 1333
removing, 1333
formation bounding volume, 1414
generating navigation data for entity, 1285
managing forces, 1331
simulation object group, 1362
starting, 1305
systems, 1305
variable binding
adding, 1488
system definition file, 1490
simulation object group, 238, 400
and selection groups, 458
connection, 1363

creating, 1362
loading with terrain, 170
Simulation Object Independently Tasked condition, 942
Simulation Object parameter, 762, 858
simulation object type
script availability, 766
script validity, 767
Simulation Object variable, 943
simulation ping time, 200
Simulation Selection Toolbar, 135
Simulation Terrain parameter, 230
simulation time, 121
Simulation Time condition, 942
Simulation Time Panel, 120
Simulation Time Scale Toolbar, 135, 260
Simulation Time Toolbar, 120, 135
simulationModelSet parameter, 1675
Simulation-Model-Set-Files parameter, 354
simulationObjectEditor.csv, 1330
simulation-run-duration parameter, 272
sink point, 416
creating, 418
SISO Enumerations Document, 119, 1562
SISO Reference for Enumerations for Simulation Interoperability, 1469
site ID, 115, 156, 176-177, 183, 189, 357, 1150, 1675
configuring, 1675
specifying at startup, 189
siteId parameter, 1675
sitting, human posture, 892
situational awareness, 1171
size
sea spray, 1108
splash effect, 1108
water droplet, 1108
window, 1603
Skill Level set data request, 900
skill level, hit probability, 1538
skipping task, 566, 572
sky
cloud cover, 324, 333
enabling and disabling, 1100
star configuration, 90
skybox cube map, 1096
omitting clouds, 1096
slope-avoidance-factor parameter, 508
slot
adding, 1442, 1444, 1451
configuring, 1444, 1451
embarkation, 1442, 1569
exclusion, 1449
ingress and egress technique, 1450
name, 682, 1442
parameters, 1446
passenger and driver, 683
type, 682, 1442
smallCompass keyword, 1180, 1613
smoke
creating, 640
plume, 1057
trail, 1051
visibility, 1072
smooth period, 214
smoothing, 213
SMS, See simulation model set
sms-filename parameter, 271
snap to grid, formation, 1416
snapshot, 264
automatic deletion, 262
clearing list, 268
creating, 267
deleting, 268
periodic, 267
rolling back to, 262
saving as checkpoint, 265
snow, 324, 333
blowing in wind, 337
depth, 329
intensity, 328
weather zones, creating, 337
SO file, 157
SOEUILayoutSettings, 1307
SOG, See simulation object group
soil type
changing drag and traction factors for ground vehicles, 506
configuring for GDB, 1263
mapped to roughness type, 505
point and feature, 1262
soil-factors parameters, 505
soil-list parameter, 505
solid line, 1033
sonar
active and passive, 310
active mode, 866
depth, 900
dipping, 706
information, 463
sonobuoy, 705
thermocline, 310, 347
Sonar Depth set data request, 900
Sonar Dip task, 706
sonobuoy, deploying, 705
sorting Task and Set menus, 1640

sound effects, 1142
  decibel, 1144
  enabling, 1142-1143
  mapping to entities, 1145
space entity
  measurement unit, 1634
Space Follow mode, 1217
spawn burst, 421
spawn event, scheduling, 419
spawn interval, 420-421, 424
  variance, 420-421, 424
spawn pattern, 419
  bursts, 424
  copying, 426
  default, 422
  deleting, 427
  editing, 426
  scenario-specific, 422
  timing, 424
spawn point, 416
  activating and deactivating, 418
  creating, 417
  enabling and disabling, 1038
spawned entity, 416
specifying
  altitude, 869
    fixed-wing entity, 510
  bounding volume, 1377
  damage model, 1392
  FED file, 192
  frame rate mode, 354
  heading
    manually, 389
    set data request, 882
  HLA federation execution, 190
  language, 112, 170, 187, 189, 1306
  MOPP level, 922
  movement model, 1392
  notification level, 173, 182, 189, 1149, 1308, 1672
  pitch, 389
  platform, 1342
  point-to-point and multicast modes, 192
  port at startup, 192
  simulation model set, 183
  target, 905
  unit target, 924
specular
  color, 1089
  map, 86
speed, 470
  aircraft, 464, 473
  gain, 1201
  scaling, See speed scaling
  specifying, 891
  specifying measurement unit, 1634
  tidal current, 341, 345
  wind, 324
speed scaling, 1200
  enabling and disabling, 1200
  mode, 1201
speed-to-rpm-factor parameter, 311
SpeedTree, 90
  license, 92
  LOD, zoom, 168
  planar reflection, 1093
  shadow, 1102
  synchronization, 168, 323, 342
spheroidSemiMajor parameter, 1675
spheroidSemiMinor parameter, 1675
spider web
  bullseye, 1028
  drawing, 1015
splash effect, 1106
  size, 1108
spot report, 167, 290, 739
  3D models, ground truth, 295
  custom viewpoint, 292
  decay rate, 294
  element definition
    merge, 171
  enabling and disabling, 290
    for simulation object, 901
  icon
    changing, 1351
    updating, 1349
  label, 296
  leaf file
    merge, 171
  role in target detection, 304
  setting defaults in sysdef file, 293-294
  tactical graphics, for, 295
  track history, 296
    pinning, 297
  using in task, 296
  viewpoint, 291-292
Spot Reports set data request, 901
spotlight, 1090
spot-report-generator.sysdef file, 293-294
spray effect, enabling and disabling, 1110
spreadsheet, tags, 1331
SQL database, simulation event logging, 88
squad
  dismount, 670, 909
  dismount task, 691
  mount, 672, 910
  simulation object icon, 1080

Squad Travel to Waypoint task, 663
squatting, human posture, 892
squawk indicator, 1073
  color, 1076
  enabling and disabling, 1074
  lifetime, 1076
standard beam, 878
standard terrain damage, 1560
standard-power-type parameter, 1560
standard-terrain-damage-table, 1561
standard-terrain-damage-table block, 1560
standard-terrain-damage-type parameter, 1560
standing, human posture, 892
stars, configuration, 90
Start/Resume message, 273
starting
  back-end, 129
  front-end, 129
  scenario
    event, 283
    load-balanced, 258
  simulation, 260
  Simulation Object Editor, 1305
  VR-Forces, 129
startInRunMode parameter, 1675
startMinimized parameter, 1675
startup
  back-end minimize, 1675
  option, 163
  tutorial, 134
state
  sea, 344
  sending update, 1172
  simulation object information, 463
state property, 1490
  accessing from lua, 1493
  adding, 1490
  data types supported, 1492
  displaying in Information dialog box, 1493
  user-defined, 1490
state repository, 1684
  hierarchy, 1474-1476
statement
  in plan, 949, 955
    drag and drop, 981
state-repository parameter, 1684, 1686
station keeping task, 595
statistics
  displaying, 202
  VR-Forces performance, 202
status
  combat engineering object, 536
  entity, 481
  navigation area, 1283
  personnel, ammunition, weapons, and POL, 465
  subtask, 796
  task, 565, 796
statusUpdateHeartbeat parameter, 1675
stealth
  control, 975
  mode, 907
  window, 137
Stealth observer mode, 126, 1164-1165
stencil bits, 174
stencil buffer, 207
step-function, 1551
stereo display types, 1616
stereo, polarized, 1617
Stop/Freeze message, 273
stopping
  movement, 733
  scenario event, 285
  subtask, 797
  task, 566, 797
stopping-distance-factor parameter, 1568
stopping-factor parameter, 507
stopSubtask() function, 797
stopTask() function, 797
stores, restoring for simulation object, 896
Stow Hand Item set data request, 902
Stow Port Refueling Drogue task, 701
Stow Refueling Boom task, 700
Stow Starboard Refueling Drogue task, 701
strafe ground target, 628
strategic missile, 319
strategic missile weapon system, 639
streaming
  terrain, 1248
street light, 1090
strength, 528
strong point
  creating, 725
  improving, 729
style
  fill, 1033
  line, 1033
  pen, 1033
subcomponent, 1683
  classes, 1473
subgoal, 1274
submarine, See subsurface entity
subnet mask, 177, 184, 1679
subnetMask parameter, 1679
subordinate
  adding to unit, 1408
  editing, 1410

in condition, 963
information, 465
order, changing, 1410
removing from unit, 1409
replacing, 1409
task status, 942
subsurface entity
lowering periscope, 702
measurement unit, 1634
moving to depth, 606
patrol area task, 609
raising periscope, 702
RPM, 310
subsurface-entity-param parameter, 1476, 1688
subsurface-entity-state-repository, 1476
subsystem, See also system
Subsystem Enabled set data request, 902
subtask
Lua, 795
state
canceled, 796
complete, 796
running, 796
status, 796
stopping, 797
subtaskResult() function, 797
sun rays, 1095
sunbeam, 1095
superior
specifying, 902
tasking simulation object, 906
Superior set data request, 902
super-type, 1470
supplies, 537
information, 465, 536
providing, 539
receiving, 538
starting or stopping delivery, 926
supplies parameter, 1438
supply line, 1071
Supplying set data request, 926
support point, specifying, 1379
suppressDataAvailableException parameter, 1675
suppression-degrades-sensor-performance-by parameter, 522
suppression-detonation-insult parameter, 522
suppression-detonation-range parameter, 522
suppression-insult-level parameter, 521
suppression-recovery-time parameter, 521
suppressive fire
at location, 520
at point, line, or area, 520, 637, 645
configuring, 521
Suppressive Fire task, 678
suppressSelfReflect parameter, 1679
surface
changing drag and traction factors for ground vehicles, 506
clamping to, 1248
damage specification, 1551
transparency, 341, 345, 1611
type, mapped to roughness type, 505
surface attack, 630
surface entity
buoyancy, 1111
measurement unit, 1634
patrol area task, 609
RPM, 310
wake, 1051
wake configuration, 1395
zig zag to heading task, 622
Surface Vessel Lighting set data request, 903
surface-entity-param parameter, 1476, 1688
surface-entity-state-repository, 1476
surge depth, 333, 341, 345
surrender
condition, 941
set data request, 903
Surrendered set data request, 903
suspend() function, 788
swapping simulation object systems, 1393
swell, sea, 86, 333, 341, 344
switch node
displaying, 1353
visual model, 1352
Switch Node States, 1352
switching observer mode, 1165
symbol decoration
LOD out Altitude, 475
position, 475
showing and hiding, 473
symbol, location of, 103
Synchronization Matrix
building, 986
entity, 987
function, 984
new, 986
object, 985
open, 986
overview, 984
phases, 987
plans, 988-990
units, 985, 987
synchronization, vertical, 216
Synchronize Laser Code set data request, 305, 903

synchronizing
back-ends, 142
data and settings, 126
laser codes, 903
sysdef file. See system definition file, 1482
system, 1482
active-radar-sensor, 540
adding to simulation object, 1391
aggregate-comms-jammer, 540
aggregate-radar-jammer, 541
connection, 1483
damage, 1546
embedded entity, 411
emissions-sensor, 541
entity, 1390
disabling, 902
enabling, 902
status, 463
explosive device, 1536
marker in 3D model view, 1392
missile launcher, 1532
movement properties, 1433
Naval Depth Charge Deployment, 625-626
Naval Mine Deployment, 626, 628
naval mine explosive device, 1536
property, editing, 1392
Remote Sensor Receiver, 895
removing, 1392
replacing on simulation object, 1393
script, 745
scripted task, creating, 779
sensor, 1504
detection probability table, 1510
Simulation Object Editor, 1305
simulation object, editing, 1390
tactical radar jammer, 306
weapon, 1520
system definition file, 1482, 1520
applying new variable bindings, 1326
configuring connections, 1483
new, 1482
sensor, 1505
specifying scripted task in, 768
variable binding, 1486
adding, 1487
adding to Simulation Object Editor, 1490
system script directory, 178
system scripted task, adding to menu, 780

T

tab
overlay, 1006
tabbed panel, 1622
table
detection, 303
Lua, 796
taskParameters, 797
TACAN transmitter set data request, 904
tactical air navigation, transmission, 904
tactical data link
message, 298
track history, 300
message display configuration, 299
messaging set data request, 907
settings, 299
tactical formation, 1411
tactical graphic, 994
active and inactive, 1035, 1038
attaching to entity, 871
attaching to simulation object, 1028
automatically attaching to simulation object, 1029
box, drawing, 1016
bullseye, 1025
spider web, 1028
changing name, 393
changing state with set data requests, 1038
color, 1033
compared to combat engineering object, 536
console, notification level, 483
creating, 372-374, 377, 381-382, 1012
deleting, 1029
editing, 1029
element definition
prevent loading, 172
export to shapefile, 1008
extruded, 1016
force, 1040
HAT lines, 1000
hiding from Overlays tab, 999
hier file
directory, 174
merge, 171
importing shapefile to, 1009
information, 462
label, 1000
labels, 995
leaf file
directory, 174
merge, 171
loading, 1007
menu icon, 1364
model, 1366
moving to different overlay, 1017
notify level, 1041
performance, 1000

placing, 382
property
default value, 1364
editing, 1365
keyword, 1365
setting at creation, 384
publishing, 1019
range, 1026
saving, 1007
selecting, 455
showing and hiding, 998, 1035, 1038
texture, 1366, 1368
type map
merge, 172
prevent loading, 172
vertex
adding, 1030
deleting, 1032
volumetric, 1016
weather, 339
Tactical Graphic Active condition, 942, 1035, 1038
tactical graphic creation palette, 373
tactical graphics creation palette, 374
tactical radar jammer system, 306
tactical smoke
dissipation, 640
enabling and disabling, 1072
launching, 640
tactical-smoke-param parameter, 1477
tag
adding, 1327
assigning based on 3D model characteristics, 1330
changing, 1337
deleting, 1327
editing in spreadsheet, 1331
exporting, 1330
importing, 1331
new, 1327
object
changing, 1337
loading, 246
Object Group, 400
removing, 1329
renaming, 1327, 1329
taillight, 1090
takeoff
fixed-wing entity, 511, 619-620
rotary-wing entity, 514
vertical, 621
Takeoff (Vertical) task, 621
Takeoff On Route task, 619
Takeoff On Runway task, 620
talking, human task, 651
tank scrape, 1023
tanker aircraft
internal refueling, 692
refueling station, 587
Tanker Lighting set data request, 904
target
detecting, 1528
probability of, 1510
detection, 302
spot reports, 304
firing at, 303, 632, 635-636
identifying with laser beam, 304
jamming, 866
priority list
add to, 905-906
clear, 906
update, 906
viewing, 905
specifying, 905
with laser, 637
strafing, 628
type, 1539
unit, preferred, 924
Target Priority List - Add set data request, 905
Target Priority List - Clear set data request, 906
Target Priority List - Update set data request, 906
target selection controller, 1532-1533
Target set data request, 905
targetFrameRate parameter, 1676
targeting-control parameter, 1528
task, 558
Activate Jammer, 737
Airdrop, 683
Airdrop Along Route, 585
Airdrop from Aircraft, 684
Air-to-Air Attack, 713
Air-to-Ground Air Base Attack, 713
Air-to-Ground Attack, 714
Animated Movement, 585
Anti-ship (Fixed Depth), 641
Arm Mine at Depth, 624
articulated parts, moving, 698
Assault, 675
assigning to unit, 572
Assume Coil Formation, 674
Attack Aircraft, 625
Attack Aircraft with Guns, 625
Attack By Fire, 676, 714
Attack By Fire in Rotation, 676
Attack with Anti-Air Missile, 715
Attack with Anti-Ship Missile, 715
Attack with Land-Attack Missile, 715
Attack with Torpedo, 716

Automatic Air Defense, 716
background, 740
background process, 812
Biological Attack, 716
Bomb Location, 717
Bound In Lanes, 673
Bounding Overwatch Successive Continuous, 673
Breach Obstacles, 719
by superior, 906
C++, 745, 795
Car Door, 698
Cease Fire, 625
Change MOPP Level, 732
Change Posture, 733
Chemical Attack, 717
Circle Around Location, 650
Civilian Idle, 650
Civilian Pray, 650
Civilian Protest, 651
Civilian Talk, 651
Clear, 676
Close Car Door, 699
Close Cargo Door, 700
Close Door, 651
Close Window, 651
Come to Stop, 586
concepts, 554
concurrent, 554, 561, 750, 765, 771
configuring mutually exclusive, 580
confirmation prompt, enabling and disabling, 561
Construct Abatis, 720
Construct Anti-Tank Ditch, 721
Construct Barbed Wire, 721
Construct Barricade, 722
Construct Berm, 722
Construct Booby Trap, 722
Construct Bridge, 723
Construct Dry Gap, 723
Construct Fortified Area, 724
Construct Fortified Line, 724
Construct Minefield, 725
Construct Strong Point, 725
Convoy Along, 572, 586
Convoy To, 572, 587
Crank Left, 587
Crank Right, 587
Create More Entities, 652
Crowd Along Line, 652
Crowd Around, 652
Crowd Around Waypoints, 653
Crowd in Front of Entity, 654
Deactivate Jammer, 737
Deploy Port Refueling Drogue, 701
Deploy Refueling Boom, 700
Deploy Sonobuoy, 705
Deploy Sonobuoys Along Route, 705
Deploy Starboard Refueling Drogue, 701
Destroy Bridge, 726
Destroy Ditch, 726
Destroy Explosive, 727
Destroy Fortification, 727
Destroy Obstacle, 727
DI-Guy Animation, 654
DI-Guy Gesture, 654
Disembark, 681
Disembark All, 681
Dismount, 691
Dismount Headquarters, 669
Dismount Platoon, 670
Dismount Squad, 670
Dismount Team, 671
Disperse Crowd, 655
Drop Fuel Tanks, 738
Drop Naval Depth Charge, 625
Drop Naval Depth Charge at Location, 626
Drop Naval Mine, 626
Drop Naval Mines Along Route, 628
Drop Off Passengers at Location, 669
Embark, 407, 681
Engage From Soldier Firing Positions, 677
Engage From Vehicle Firing Positions, 677
escaping retask procedure, 563
Escort Air Unit, 733
Establish Refueling Station, 587
Execute Surface Attack, 630
Explode Charge at Depth, 631
Extend Towed Array Decoy, 702
Extract Using Ladder, 690
Extract Using Rope, 687
FASCAM, 738
Fast Rope Insertion, 686
filtering simulation object list, 565
Find Cover, 655
Find Firing Position, 656
Fire At, 632
  shortcut, 571
Fire Ballistic Gun, 632
Fire Cruise Missile, 633
Fire for Effect, 634
Fire Javelin at Target, 635
Fire Laser Guided Missile, 636
Fire Missile Launcher, 636
Fire Team Bound to Location, 657
Fire Team Occupy Fire Position, 658
Fire Team Suppress Target, 520, 637
Fire Team Travel to Waypoint, 658

Fixed Wing Land At Aerodrome, 512
Fixed Wing Land On Runway, 512
Fixed Wing Takeoff On Runway, 511
Flee From Entities, 659
Fly Altitude, 589
Fly Cloverleaf Pattern, 589
Fly Figure 8 Pattern, 590
Fly Heading, 592
Fly Heading and Altitude, 568, 591
Fly Loiter Pattern, 592
Follow Along Offset Route, 673
Follow Entity, 593
gesture, 654
Halt Movement, 733
Hover, 594
ID, 796
IFF Interrogate, 693
IFF Sweep Interrogate, 694
Improve Booby Trap, 727
Improve Breach, 728
Improve Bridge, 728
Improve Ditch, 728
Improve Fortification, 729
Improve Obstacle, 729
in plan, 949
independent, 554
Indirect Fire, 718
Indirect Fire, Provide, 644
Insert Using Ladder, 691
Keep Station, 595
keyboard shortcut, 571
Ladder Extraction, 688
Ladder Insertion, 690
Land (Vertical), 598
Land at Air Base, 734
Land at Airport, 596
Land at Airport Near Location, 596
Land On Route, 596
Land On Runway, 597
Lase and Fire Guided Missile, 637
Lase Target, 304, 637
Launch Anti-Submarine Missile (Vertical), 638
Launch Ballistic Missile at Location, 639
Launch Ballistic Missile at Target Object, 639
Launch Countermeasures, 311
Launch Flight, 730
Launch QRA Aircraft, 731
Launch Smoke, 640
Lower Periscope, 702
Lua, 795
  parameter, 797
Maneuver Along, 598-599
Mount Headquarters, 671
Mount Platoon, 671
Mount Squad, 672
Mount Team, 672
Move, 734
Move Along Route, 599
Move Along Route Retrograde, 734
Move Along Route with Actions, 600
Move Between Cover to Location, 659
Move Into Formation, 603
Move Next to Target, 603
Move To
  for plan, 604
  shortcut, 571
Move To (Plan Path), 606
Move to Altitude, 605
Move to Bullseye Offset, 605
Move to Depth, 568, 606
Move to Location, 604
Move to Location (Plan Along Roads), 735
Move to Retrograde, 735
Move to Waypoint (Plan Along Roads), 735
Movement To Contact, 677
mutually exclusive, 554, 561, 935
Nuclear Attack, 718
one click, 571, 751
Open Cargo Door, 699
Open Door, 660
Open Window, 660
Orbit, 607
Orbit Object, 607
Parachute to Target, 660
parameters, 563
Paratrooper Air Drop, 608
Patrol Area, 609
Patrol Between, 609
Patrol Route, 609
Pattern Hold, 610
performing, 366
Pick Up Passengers at Location, 669
Place IED, 643
Prepare QRA, 731
procedures, introduction, 560
Protest Along Line, 661
Protest Around Location, 662
Provide Indirect Fire, 644
Provide Mobile Overwatch, 674
Provide Stationary Overwatch, 675
Provide Suppressive Fire, 645
Provide Suppressive Fire from Cover, 520
Provide Suppressive Fire on Location, 520
Raise Periscope, 702
reactive, 436, 572, 707, 746, 789, 811
Refuel From, 610

Refuel from Tanker, 738
Refueling Mission, 739
Release Bomb, 646
Release Bomb on Laser Spot, 646
Release Bomb on Location, 646
Release Next Entity From Queue, 706
Release Unguided Bomb, 647
retask, 558
Return to Base, 611
riot, 662
Rope Extraction, 686
Rope Sling Load Drop Off, 685
Rope Sling Load Pick Up, 685
Rope Suspention Insertion, 688
Rotary Wing Airlift Cargo to Location, 611
Rotary Wing Land, 612
Rotary Wing Retrieve Cargo, 612
Rotary Wing Unload Cargo, 612
Rush to Location, 662
Sail Heading, 568, 613
scripted, 745-746
  action category, 581
  adding to toolbar, 769, 1646
  available by behavior set, 767
  concurrent, 771
  execution flow, 790
  filtering list, 776
Search and Rescue
  Fly Creeping Line Pattern, 613
  Fly Expanding Square Pattern, 615
  Fly Parallel Pattern, 616
  Sector Search, 618
Sector Search Operation, 703
Seize Objective, 678
Send Radio Set, 695
Send Radio Task, 696
Send Text Message, 697, 972
simulation object, 367
skipping, 566, 572
Sonar Dip, 706
spot report, using in, 296
Squad Travel to Waypoint, 663
state
  canceled, 796
  complete, 796
  running, 796
status, 463, 565, 796
stopping, 566, 797
Stow Port Refueling Drogue, 701
Stow Refueling Boom, 700
Stow Starboard Refueling Drogue, 701
Strafe Ground Target, 628
Suppressive Fire, 678
Takeoff (Vertical), 621
Takeoff On Route, 619
Takeoff On Runway, 620
Throw Grenade at Location, 648
toolbar, adding to, 1646
Transition Into Formation, 621
Turn to Heading
  aggregate, 736
  entity, 622
unit concepts, 518
user, 707
viewing current, 943
Visit Neighbor, 663
visualizing, 1060, 1062
Wait, 697-698
Wait Duration, 698
Wait Elapsed, 698
Wait for Escort, 736
Wait in Queue, 664
Wander, 664
Wander In Area, 665
who can execute, 562
Zig Zag to Heading, 622
Task menu
  editing, 1644
  extended name, 754
  icon, 753
  location for script, 764
  sorting, 1640
  system scripted task, showing and hiding, 780
task parameters
  pinning, 564
task visualization, 1676-1677
  pinning, 1062
Tasked by Superior set data request, 572, 906
  using in plans, 945
task-manager parameter, 1685, 1687
taskParameters table, 797
taskResult() function, 797
taskVisualizationPublicationMode parameter, 1676
TDL, See tactical data link
TDL Messaging set data request, 907
team
  dismount, 671, 909
  mount, 672, 910
tearing off menu, 1628
technique
  egress, 1450
  ingress, 1450
teleporting observer, 1189
temperature, air, 327, 333

template
unit, 1418
temporary directory, scenario, 358
terminating the application, 146
terrain, See also terrain database; terrain server
aerodromes, 1292
agility, 84, 1242
analyzing, 1269
avoidance, 514
caching earth files, 90
color, 1260
composability, 1242
composing, 1238
coordinates, 1262
damage-by-power, 1560
databases provided, 1238
destructible, 1238
drag
disabling, 1187
with mouse, 1187
dynamic, 165
damage, 1560
global dynamic terrain processor, 232
man-made, 1023
navigation data regeneration, 1284
embedding
buildings, 401
cultural feature, 401, 1381
extracting props from, 1244
formats, 1243
GDB soil type, 1263
geocentric, 1246
intersection, 221
large, 1246
limit textures, 169
loading
from command line, 174
simulation object group, 170
magnification angle, 1193
opening, 1246
orbiting with mouse, 1189, 1191
paged
loading, 181
paging, 1247
patch, 1243
placement of new entities, 386
profile. See terrain profile, 1266
scale factor, 1192
shape, 1243
soil type, 1262
streaming, 1248
supported formats, 1243
wireframe mode, 1254
terrain damage
standard, 1560
terrain database, 84, 1245
Aeronautical Charts, 1239
Ala Moana, 1238
Arrivals Hall, 1238
Bakersfield topo map, 1238
Brooklyn, 1238
changing for a scenario, 356
configuring, 1676
path to, 1670
Destructible Terrain, 1238
loading, 144
at startup, 184
MAK Earth, 1238
NTC (online), 1239
provided, 1238
Sadr City, 1239
Truman carrier, 1239
VR-Village, 1239
Terrain Magnification High LOD Angle, 1193
Terrain Magnification Scale Factor, 1192
terrain performance, 221
terrain profile, 1267
concepts, 1266
configuring, 1267
line, 994, 1018, 1269
opening automatically, 1267
sampling frequency, 1267
terrain profile line, 1269
track history, 1057
vertex, 1267
Terrain Profile window, 1266-1267
terrain server, 1246, 1248
terrain-check parameter, 513
terrainDatabase parameter, 1676
Terrain-Database parameter, 353
TerraPage file, 1243
Test Result command, 979
testing condition in plan, 939
Tether Location Track mode, 1219
Tether mode, 1197
Tether observer mode, 1182, 1214, 1218
Tether Track observer mode, 1219
text
in title bar, 1633
label color, 474
left-to-right, 167
message (command), 972
message (task), 697, 972
object, resizing, 1032
shadow, 476
tactical graphic, 1017

text editor, 746
  specifying for scripts, 781
texture
  earth file
    limit splatting, 168
    limit terrain, 169
    limit vegetation, 169
  entity
    limit exclusions, 169, 1307
    limit mipmap, 169, 1307
  map, 86
  ocean planar reflection
    height, 1093
    width, 1093
  sharing, 204
  tactical graphic, 1366, 1368
thermocline
  settings, 347
  sonar, 310
this, Lua object, 785
thread
  callback, 182
  disable for parallel algorithm, 185
  network callback, 182
  priority, 183
threat ring, 1065
threshold
  data updates, 1670
  DIS heartbeat, 1678
  object timeout, 1678
Throw Grenade at Location task, 648
tick
  disable parallel, 180
  publishers, 184
tick rate
  adaptive, 217
  tuning, 206
tick() function, 787
tickPublishersDuringEntityTick parameter, 1676
tidal current
  direction, 345
  speed, 345
  speed and direction, 341
tidal stream
  wake, 345
time
  advancement, 121
  evaluating in plan, 942
  format, 332
  scenario
    end, 268-269
    ending, 232
  setting, 331-332
  simulation, 120, 236
  zone, 237, 332
    setting, 332
time constraints, 142
time management
  concepts, 142
  configuring, 143
  enabling, 143
    for sim.engine, 144, 186, 192
  RTI requirements, 143
  setting simulation speed, 260
Time Multiplier parameter, 232
time of day, 120-121, 330
  advancement, 121
  illumination, 1515
time scale
  parameters, changing automatically, 261
  range on Simulation Time Scale Toolbar, 261
time stamp, See timestamp
time zone, 330
time, regulating, 142
time-based events, synchronize, 168, 323, 342
timed fuse, 875
time-multiplier parameter, 353
timeout
  DIS, 186
  interval, 176
  object, 1678
  track history, 1055
timestamp, 166
  absolute
    enable, 150, 152, 176-177, 184
  order, configuring, 1678
  ordered message, 143
timing, spawn pattern, 424
tire track, 1051, 1053
title bar, adding text, 1633
TLE file, 246
toolbar
  adding task or set, 1646
  aerodromes, 1292
  button, 135, 769, 1646
  configuring, 1640, 1645
  displaying and hiding, 1624
  docking, 1623
    vertically, 1623
  icon, specifying, 1646
  locking, 1647
  MAK radio, 1153
  Objects, 440, 456
  scheduled event, 276
  set, adding script, 769
  task, adding script, 769

undocking, 1623
toolbar element, 1646
toolbarrow element, 1646
toolkit
API, 99
SDK, 99
topological model, 123
torpedo, 716
anti-submarine, 638
launching, 641
tracer
ammunition, 1394
configuring, 1529
displaying, 1057
using, 907
Tracer Use set data request, 907
tracer-type parameter, 1529
track history
altitude, displaying, 1056
configuring, 1055
displaying, 1054
enabling and disabling for all sim. objects, 1054
length, 1055
spot report, 296
pinning, 297
tactical data link message, pinning, 300
terrain profile, 1057
timeout, 1055
Track observer mode, 1182, 1214, 1219
track, radar mode, 894
tracking
ammunition, 915
beam, 878
entity resources, 481
equipment, 919
resource, 925
weapon, 927
Tracy Client, 203
enable, 166
traffic
generating, 416
spawn pattern, 419
trailing effects, 1051
enabling and disabling, 1051
model, 1052
simulation object, 1051
trajectory
history, See track history
indirect fire, 893
smoothing, 213
transient intervisibility
displaying, 1125
object, 1124
Transition Into Formation task, 621
transition time, posture
editing, 1434
urgency, 908
translation, VR-Forces GUI text, 108, 1440
transmitter, publishing radio, 1574
transparency
2D icons, 479
alpha to coverage, 207
surface, 333, 341, 345, 1611
transponder
emergency, 878, 1391
IFF, 883
trap, 722
Travel posture, 530
tread track, 1051, 1053
triangle icon, 486
triangle, vertex label, 1000
trigger, 935
adding, 951
automatically checking for unused, 954
cleaning up unused, 954
name, 936
plan order, 944
reactive task, 572
registration, 937, 953
taskless, 944
unregister, 953
Triton Ocean SDK, 91
Triton.config
ocean configuration file, 1194
troubleshooting, 89, 1659-1662
true airspeed, 464, 473
Truman carrier terrain, 1239
TSO, See timestamp, ordered message
tuning
network performance, 212
performance, component, 206
visual performance, 212
Turbosquid, 92
turbulence, air, 340
turn rate, 591-592
Turn to Heading task
aggregate, 736
entity, 622
turret, attaching observer to, 1212
tutorial
embarkation slot configuration, 1453
plan
complex condition, 958
simple condition, 956
starting VR-Forces, 134
two-dimensional view layer, 1262

TXP file, 1243
type
  keyword for translation, 1440
  script parameters, 755
  slot, 682, 1442

U

UI file, 1488-1489
unconstrained time, 142
undercarriage inspection system, 1295
underFireDistance parameter, 897
under-fire-parameter parameter, 941
underFireTime parameter, 897
Underwater Acoustics PDU, 310
underwater, visibility, 341, 344
Undetectable set data request, 907
undo ellipse rotation, 1032
undocking
  menu, 1628
  panel, 1622
  toolbar, 1623
undoing
  category changes, 1329
  simulation object edits in SOE, 1345
unexploded ordnance
  aggregate-level modeling, 535
  destroying, 727
  tactical graphic, 1021
unique ID, 363
  UUID, See UUID
unit, 1687, See also simulation object
  adding simulation object to, 440, 902
  aggregate-level, 440
  ammunition, weapons, equipment, 1420, 1422
  artillery, 718
  assign vehicle, 909
  assigning tasks, 572
  attrition, 535
    from combat engineering object, 535
  bounding volume
    color, 450
    displaying, 448
    Simulation Object Editor, 1414
  Circle Around Location task, 650
  closing formation, 517
  collapsing, 444
  collapsing to root, 444
  convoy, 586-587
    behavior, 572
  count of, 487
  creating, 438, 1407, 1418
    preconfigured, 440
  creating in Order of Battle Panel, 493
  crowd, 432
  deleting, 442
  dismount
    headquarters, 909
    HQ, 669
    platoon, 670, 909
    squad, 670, 909
    team, 671
  displaying ghosted icons, 446
  echelon ID, 516
  editing, 1407
  element definition
    prevent loading, 173
  embark task, 681
  empty, 1407
  expanding, 444
  footprint, 529
  formation, 517, 879, 1566
    adding, 1412
    automatic layout, 1415
    copying, 1414
    default, 1413
    editing, 1411
    manual layout, 1415
    removing, 1413
    renaming, 1413
    snap to grid, 1416
  herd creation, 433
  hier file
    directory, 174
    merge, 171
  icon, displaying, 448
  information, 462, 465
  leaf file
    directory, 174
    merge, 172
  mount
    headquarters, 910
    HQ, 671
    platoon, 671, 910
    squad, 672, 910
    team, 672, 910
  movement, 517, 531, 598-599
    affected by combat engineering objects, 533
  moving to formation, 603
  new, 438, 1407, 1418
  organization concepts, 516
  parameter type, 1501
  posture, 530
  preconfigured, 440
  removing simulation object from, 441, 902
  reorganizing, 516-517, 895

restoring, 896
sector, 529, 1065
selecting, 440, 444, 457
sensor, 533
setting formation in plan, 945
specifying as Aggregate As option, 1406
subordinate
adding, 1408
changing order, 1410
editing, 1410
functions, 1410
in condition, 963
removing, 1409
replacing, 1409
target, preferred, 924
task behavior, 518
task status of subordinates, 942
template, 1418
transition to formation, 621
type map
merge, 172
type mapping file
prevent loading, 173
writing plan for, 980
Unit Patches set data request, 908
units of measurement, 1634
Universal Transverse Mercator, See UTM coordinate system
Unloaded set data request, 911
unloading
cargo, 612
entities from entities, 406
unlocking overlay, 1005
unpublishing tactical graphics, 1019
unregistering a trigger, 953
unselecting
object, 457
simulation object, 457
unused range, 1548
unused trigger
checking for automatically, 954
cleaning up, 954
update
frequency of, 1670
messages, 1172
rate, intervisibility, 1127
request
disable, 175
on initialization, 176
Update_EW_Degradation script, 540
updating
platforms, 1326
target priority list, 906
visual models, 1360
weapon and damage systems, 1322
window layout, 1627
Urgency set data request, 908
useAbsoluteTimeStamps parameter, 1676
useAdvisories parameter, 1678
useAsyncIO parameter, 1679
useCustomMapDatum parameter, 1676
USE-DEFAULT keyword, 358
useDrawControlForTaskVisualizations parameter, 1677
useIpv6 parameter, 1679
use-logger-control parameter, 271
user
data, specifying directory, 174, 184
scenario, specifying directory, 174, 254
settings, 125
task, 707
terrain, specifying directory, 174
user-defined formation for unit, 1567
user-defined state property, 1490
useSingleNetworkThreadWhenRtiThreadSafetyNotGuaranteed parameter, 1678
UTM coordinate system, 1182, 1245, 1635
UUID, 264, 362
scheme, 173, 182
simulation object, 363

V

vapor trail, 1051
variable
binding, See variable binding
Created Object, 943
global, saving state, 794
plan, 943
Simulation Object, 943
tick rate, 206
variable binding, 1486
applying changes, 1326
default value, 1487
new, SOE, 1488
platform file, 1486-1487
system definition file, 1486-1487
variable frame mode, 215
variable, environment, See environment variable
variable-frame keyword, 354
variable-frame run-to-complete, 122, 354
variance
heartbeat, 1679
spawn interval, 420-421

vector
  geocentric, 803
  Lua, 801, 803
  network, 503
vector feature
  configuring avoidance of, 1568
  list, 1258
Vector3D class, 801
VectorGeoc3D class, 803
VectorOffset3D class, 803
vehicle
  displaying drivers, 1085
  hiding a driver, 882
  model, 79
  unit assignment, 909
velocity
  displaying, 470
  measurement unit, 1634
version
  displaying, 112, 175, 185, 1151, 1309
  plug-in
    checking, 166
    skip check, 166
  product, 96
  RPR FOM, 151, 176, 186, 1150, 1678
version parameter, scenario file, 355
vertex
  adding, 1030
  altitude, 1000
  deleting, 1032
  editing, 1031
  line, terrain profile, 1267
  showing and hiding, 1000
vertical landing task, 598
vertical sync, 217
vertical takeoff task, 621
video card, processor speed, 199
video, adding to scenario event, 279
view
  adjacent channels, 1661
  clearing list, 1204
  constraint
    disabling, 386
  constraints, 1200, 1214
  default, 1205
  deleting, 1204
  dragging with mouse, 1189
  embarkation configuration, 1443
  floor, 1200
  go to location, 1207
  inset, 1224
  loading saved, 1206
  model, 1353
  multichannel, 1613
  navigating, 1177
  observer, transitioning between, 1204
  other MAK 3D GUI viewers, 1172
  panning, 1189
  recalling, 1202
  restoring, 1204
  saved with scenario, 225, 350
  saving, 1202, 1206
    individual, 1206
  sensor, 1228, 1230
  zooming, 1191
    2D, 1192
    3D, 1191
view control message, 1207
  enabling processing of message, 1208
  script, 1207
  sending in a plan, 975
view mode
  Absolute, 1215
  Follow, 1215
  Mimic, 1215
  Mimic Location Track, 1217
  Mimic Track, 1216
  Space Follow, 1217
  Tether, 1218
  Tether Location Track, 1219
  Tether Track, 1219
  Track, 1219
viewing
  Logger files, 274
  obstructed terrain in terrain profile, 1129
  plug-in list, 162
viewpoint
  spot report, 291-292
    custom, 292
viewport, 1604
  changing, 1610
visibility, 324, 327
  configuring at different elevations, 324
  modifier, 535
  observer, 333
  of observer to other applications, 1172
  underwater, 333, 341, 344
  water, 1611
visible surface, configuring, 1263
Visit Interest Points reactive task, 436
Visit Neighbor task, 663
visual definition
  conflict, 1317
  reconcile, 1317
visual effect, displaying, 1354

visual marking
editing, 888
randomized, 1357-1358
specifying, 1357
viewing, 1358
visual model
adding, 1351
details, 1352
editing, 1346
exporting, 1359
importing, 1359
pruning, 1359
refreshing, 1360
tactical graphic, 1366
visual quality
configuring, 213
visual sensor, 1515
gimbaled, 1228
visual type, changing, 1356
visualizing tasks, 1060, 1062, 1676-1677
volumetric tactical graphic, 1016
vrf class, 785
vrf-aggregate-state-repository, 1477
vrf-base-object-param parameter, 1501, 1682
vrf-base-object-param parameter type, 1687
vrfGui, 114
vrfGui.log file, 162
vrfLauncher
combined mode, 188
special command-line arguments, 188
vrfMenuIds.h header file, 1646
vrf-moving-object-state-repository, 1477
vrfMsgDump, 89
vrf-object-param parameter, 1477, 1687
vrf-object-state-repository, 1477
VR-Forces
entity or object, 120
Launcher, 129
applications, 154
starting, 129
viewing other, 1172
window, 135
VR-Forces Developers Guide, 104
vrf-overlay-object-state-repository, 1477
vrfSim, 114
daemon, starting as, 180
log file, 182
running as daemon, 1669
vrfSim.log file, 162
vrfSim.mtl, 1280
advisory messages, 1678
vrfSim.mtl file, 1668
vrfSim.opd
remote-configurations block, 1593
required in every SMS, 1314
vrfSimSettings.xml file, 212
vrf-tactical-smoke-state-repository, 1477
vrfutil functions, 787
VRV_ALPHA_TO_COVERAGE_RENDERBIN environment variable, 208
VR-Vantage Control Toolkit, 1207
VR-Vantage view control, 975
VR-Village terrain, 1239
VSync, 166, 216-217
VTOL
land, 598
takeoff, 621
vulnerability, 528
modifier, 534

W

wading, human posture, 892
Wait Duration task, 698
Wait Elapsed task, 698
Wait for Escort task, 736
Wait in Queue task, 664
Wait task, 697-698
Wait Until statement, 938
in plan, 951
wake, 86, 91, 1051
enabling and disabling, 1110
model definition, 345
surface entity configuration, 1395
tidal stream, 345
wakeLength parameter, 345
walking, human posture, 892
wall clock time, 121, 143
wall, not moving through, 1200
Wander In Area task, 665
Wander task, 664
warfare
electronic, 306, 539-541, 895
model, aggregate, 528
warhead actuator, 1536
warhead parameter, 1529, 1546
warhead-detonation-actuator, 1536
warning message, 1672
water
droplet size, 1108
pacing and tracking, 921
splash, 1106
visibility, 1611

watercraft
lighting, 903
waypoint, 996
water-slide-depth parameter, 507
water-slide-factor parameter, 507
wave
breaking, 347
choppiness, 341
effect, 1106
spectrum, 346
wavelength, laser, 886
waypoint, 994, 996
altitude, 1000
civilian visit point, 436
crowd gathering at, 653
on surface entity, 996
patrolling between, 609
riot, 662
weapon
anti-air, 716
direction fire, configuring, 1527
fixed, configuring, 1536
laser guided, 637
pacing and tracking, 927
status, 465
system, 1390-1391
system definition file, 1482
unit, 1420, 1422
weapon interface, 1532-1533
weapon system, 1520, 1527
bomb bay, 1536
indirect fire, 1534
migrating to new release, 1322
preview, 1389
strategic missile, 639
target configuration, 1539
Weapons Pacing/Tracking set data request, 927
weather, 333
area, See weather zone
global, 232
measurement unit, 1634
setting, 324
tactical graphic, 339
weather area, See weather zone
weather elevation layers, 324
weather zone, 336
creating, 337
environment conditions, 338
priority, 336
wheel, rotation, 1078
When statement, 933, 935
building, 956
While statement, 933, 937
in plan, 950
width
changing for line, 1033
sensor contact line, 1065
wildcard, 1471
wind, 86
corridor, 339
direction, 333
direction and speed, 324
elevation, 325
offset, 314
speed, 333
speed and direction, 324
window
adding, 1598
attributes, editing, 1603
channel, new, 1601
closing, 651
location, changing, 1603
new, 137
observer, 1167
opening, 660
printing, 138
property, 1603
removing, 1602
secondary, 137
selecting simulation objects in, 455
size, changing, 1603
specifying, 168
Terrain Profile, 1266-1267
types, 1598
window layout
changes, saving, 1627
choosing, 1625
creating, 1624
default
clearing, 1626
setting, 1626
deleting, 1628
restoring factory, 1628
reverting, 1627
updating, 1627
Window Name attribute, 1603
Window Stays on Top attribute, 1603
Window Type attribute, 1603
Windows
default install directory, 97
product installation, 96
wireframe mode, 1254
world coordinates, 1177
writing
entity plan, 948

global plan, 966
plan
for multiple simulation objects, 980
remote entities, 980

X

XML
file, plug-in, 158
deleting, 161
RTI configuration file, 107
XR mode, 1164-1165, 1171
XR scaling, 1082
xtr file, 358

Y

yaw, 1177
yellow triangle icon, 303, 486

Z

Z Far, 1604
Z Near, 1604
ZeroBrane Studio, 746, 843
Z-fighting, 165, 1214, 1606-1607, 1611
Zig Zag to Heading task, 622
zone
engagement, 305
local weather, 336
time, 237
zoom
change from 3D to 2D, 1164
LOD center, 1195
magnification angle, 1193
terrain, 1193
observer, 1191
2D, 1192
3D, 1191
ocean, display issues, 1194
performance impact, 1193
scale factor, terrain, 1192
scenario on load, 1195
sensors, 899
SpeedTree LOD, 168
terrain, scale factor, 1192
to entity extents, 1194
to extents, 1194



MAK TECHNOLOGIES 10 FAWCETT STREET, SUITE 204, CAMBRIDGE, MA 02138 USA | +1 617.876.8085 | www.mak.com