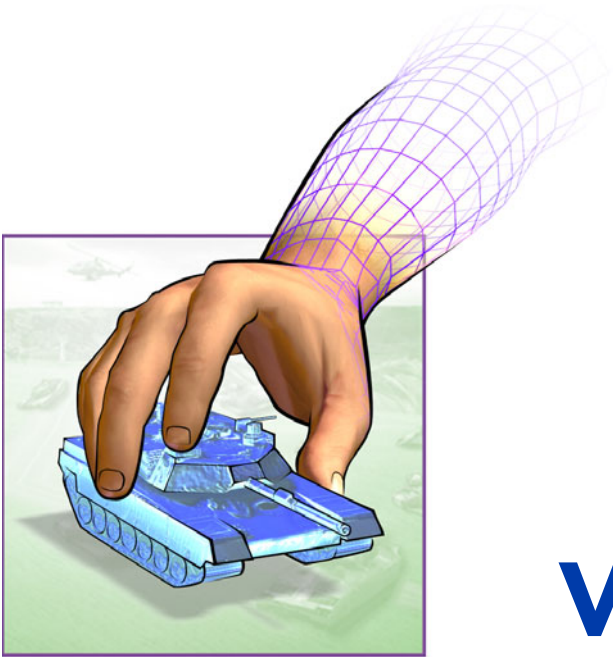
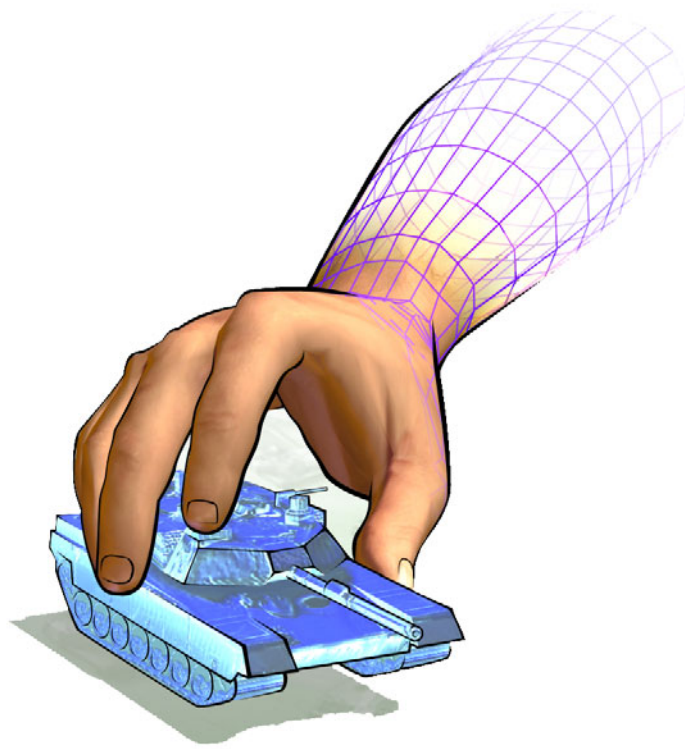




VR-Forces

Configuration Guide



VR-Forces

Configuration Guide

Copyright © 2008 MÄK 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 MÄK Technologies, Inc.

VR-Exchange™ is a trademark of MÄK Technologies. MÄK Technologies®, VR-Forces®, RTIspy®, and VR-Link® are registered trademarks of MÄK Technologies, Inc.

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

OpenFlight and Metaflight are trademarks of Presagis, Inc.

All other trademarks are owned by their respective companies.

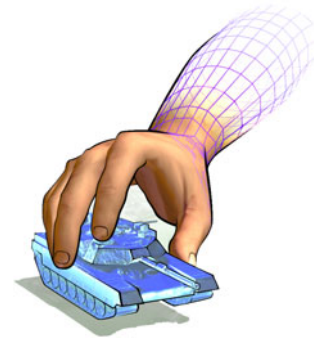
For third-party license information, please see “Third Party Licenses,” on page xix.

MÄK Technologies
68 Moulton St.
Cambridge, MA 02138 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com
www.mak.com

Revision VRF-3.11-7-080104



Contents

Preface

How This Manual is Organized	xiii
VR-Forces Documentation.....	xiv
MÄK Products	xv
How to Contact Us	xvii
Document Conventions	xviii
Hardware and Operating System Naming Conventions	xviii
Mouse Button Naming Conventions.....	xix
Third Party Licenses	xix

Chapter 1 Optimizing Performance

1.1. Introduction	1-3
1.2. Adjusting Refresh Rates	1-4
1.2.1. Changing the Frame Rate for the Display	1-4
1.2.2. Changing the Network Frame Rate	1-5
1.3. Monitoring Front-end and Back-end Frame Rates	1-5
1.3.1. Viewing Frame Rate Statistics	1-5
1.3.2. Configuring Statistics Reporting	1-7
1.4. Tuning GUI Performance	1-8
1.4.1. The Meeting Frame Rate Targets Level	1-8
1.4.2. The Not Meeting Frame Rate Targets Level	1-8
1.4.3. The Heavy Network Traffic Level	1-10
1.5. Optimizing VR-Forces 3D Front End Performance	1-11
1.5.1. Loading Entity Models and Special Effects	1-11
1.5.2. Filtering HLA Traffic	1-12
1.6. Miscellaneous Performance Configuration Options	1-12
1.6.1. Turn Off Track Histories	1-12
1.6.2. Turn Off the Display of Polygons	1-12
1.6.3. Limit Use of Object Transparency	1-13

1.6.4. Limit Use of Spot Reports	1-13
1.6.5. Configure Hazard Cloud Graphics	1-13
1.6.6. Configure Intervisibility Intersection Testing	1-13
1.6.7. Using Asynchronous I/O for DIS Exercises	1-13
1.7. Setting the Tick Rate	1-14
1.7.1. Specifying the Tick Rate for Components	1-14
1.7.2. Tuning the Network Interface	1-15
1.7.3. Tuning the State Repository Tick Rate	1-15
1.8. Setting Thresholds for Entity Updates	1-15
1.9. Tuning the VR-Forces Sleep Time	1-17
1.10. Configuring Spatial Subdivisions	1-17

Chapter 2 VR-Forces Configuration Files

2.1. MÄK Technologies Lisp (MTL) Files	2-2
2.1.1. Using Environment Variables in an MTL File	2-2
2.1.2. Specifying Lists of Lists	2-3
2.1.3. Using Conditional Statements	2-3
2.2. Configuration File Descriptions	2-4
2.2.1. The <i>vrfGui.mtl</i> Configuration File	2-5
2.2.2. The <i>vrfSim.mtl</i> Configuration File	2-14
2.2.3. Symbol Mapping Files	2-19
2.2.4. The <i>plugin.mtl</i> File	2-21
2.2.5. The <i>commonObjects.mtl</i> File	2-21
2.2.6. The <i>soundmap.mtl</i> File	2-21
2.2.7. The <i>rangeRings.mtl</i> Configuration File	2-21
2.2.8. The <i>boundingVolume.mtl</i> Configuration File	2-22
2.3. Configuring Overlay Objects	2-22
2.4. Configuring Communication Models	2-23
2.4.1. Simple Radio Communication Model	2-23
2.4.2. External Communication Model	2-23
2.4.3. Custom Communication Models	2-24

Chapter 3 Scenario Files

3.1. Introduction	3-2
3.2. The Scenario File	3-3
3.2.1. Scenario Parameters	3-4
3.2.2. Specifying Pathnames for Scenario Files	3-6
3.2.3. Changing the Terrain Database for a Scenario	3-7
3.3. Editing the Plan Files	3-7
3.3.1. The Structure of an Entity Plan File	3-8
3.4. Editing the Object Map File	3-10
3.5. Editing the Order of Battle File	3-10
3.5.1. Saving Default Entity Parameters to the Order of Battle File	3-11
3.6. The Overlay File	3-11

3.7. The Selection Groups File	3-11
3.8. Temporary Scenario Directories	3-12

Chapter 4 Merging Scenarios

4.1. The Scenario Merge Tool	4-2
4.2. Starting Scenario Merge	4-2
4.3. Creating a Scenario Merge Project	4-3
4.3.1. Adding Scenarios to a Project	4-3
4.3.2. Removing Scenarios from a Project	4-3
4.3.3. Specifying an Ignore List File	4-4
4.3.4. Saving a Project	4-5
4.3.5. Loading a Project	4-5
4.4. Merging Scenarios	4-6
4.5. Displaying Scenario Files	4-6
4.5.1. Editing Scenario Files	4-7
4.5.2. Closing Scenario Files	4-7
4.6. Specifying the Output Notification Level	4-7
4.7. Using the Scenario Merge Command-Line Interface	4-8

Chapter 5 Editing Entity Models

5.1. Introduction	5-3
5.1.1. Simulation Model Sets	5-4
5.1.2. Starting the Entity Editor	5-5
5.2. Creating a New Simulation Model Set	5-6
5.2.1. Specifying the Default Simulation Model Set for Scenarios	5-6
5.3. Opening a Simulation Model Set	5-7
5.4. Editing an Entity Model	5-8
5.4.1. Changing an Entity's Icon	5-8
5.4.2. Editing an Entity's Category	5-11
5.4.3. Editing an Entity's Force	5-12
5.4.4. Changing an Entity's Entity Enumeration	5-13
5.4.5. Changing an Entity's Name	5-13
5.4.6. Specifying Whether an Entity can be Created	5-14
5.4.7. Editing Behavioral Parameters	5-14
5.4.8. Undoing (Reverting) Changes to an Entity	5-15
5.5. Editing an Entity's Systems	5-15
5.5.1. Adding A System	5-16
5.5.2. Selecting a Damage Model or Movement Model	5-17
5.5.3. Editing a System's Properties	5-17
5.5.4. Removing a System	5-17
5.6. Creating a New Entity	5-18
5.6.1. Creating a New Entity from an Existing Entity	5-18
5.7. Deleting an Entity Model from a Model Set	5-19

5.8. Editing the Create Menu	5-19
5.8.1. Generating the Create Menu Automatically	5-19
5.8.2. Editing the Create Menu Manually	5-20
5.8.3. Undoing (Reverting) Changes to the Create Menu	5-22
5.9. Editing the Category List	5-22
5.9.1. Adding a New Category	5-22
5.9.2. Removing a Category	5-23
5.9.3. Renaming a Category	5-23
5.9.4. Undoing Category Changes	5-23

Chapter 6 Advanced Entity Model Editing

6.1. The Object Parameter Database Editor	6-2
6.2. Starting the OPD Editor	6-3
6.3. Loading an Object Parameter Database	6-4
6.3.1. Changing the Object List View	6-4
6.3.2. Loading a Create Menu Configuration File	6-5
6.4. Editing an Object's Parameters	6-5
6.4.1. Parameter Descriptions	6-5
6.4.2. Editing Individual Parameters	6-6
6.4.3. Editing Parameters for Multiple Objects	6-7
6.4.4. Deleting Parameters and Components	6-7
6.4.5. Restoring Parameters	6-7
6.5. Creating a New Object from an Existing Object	6-8
6.6. Adding New Components and Parameters	6-8
6.6.1. Adding New Components	6-9
6.6.2. Editing Components	6-9
6.6.3. Adding New Resource Parameters	6-11
6.6.4. Adding New Subordinate Object Parameters	6-11
6.6.5. Adding New Soil Type Parameters	6-12
6.7. Editing Variables	6-12
6.8. Editing Systems	6-13
6.9. Saving Changes to the Object Parameter Database	6-13

Chapter 7 The Object Parameter Database

7.1. The Object Parameter Database	7-2
7.1.1. VR-Forces Classes and Object Parameters	7-6
7.1.2. Object Types	7-6
7.1.3. Parameter Types	7-9
7.1.4. Local and Remote Subcomponents	7-9
7.2. Component Descriptors and Parameters	7-12
7.2.1. Common Elements of Component Descriptors	7-13
7.2.2. Configuring the Priority of Components	7-15
7.3. Adding a New Entity to the Object Parameter Database	7-16

7.4. Systems Definition Files	7-18
7.4.1. Creating a System Definition File	7-19
7.4.2. Configuring System Connections	7-19
7.4.3. Adding Variable Bindings	7-22
7.5. Configuring Aggregate Entities	7-23
7.5.1. Adding an Aggregate to the Object Parameters Database	7-23
7.6. Changing the Echelon ID Category and Echelon ID Level	7-26
7.7. Configuring Graphical Objects	7-27
7.8. Pathname Interpretation in Simulation Model Set Files	7-27
7.9. Updating Object Parameter Database Files for New Releases	7-28

Chapter 8 **Configuring Object Behaviors**

8.1. Configuring Sensors	8-2
8.1.1. Configuring a Sensor in an OPE File	8-2
8.1.2. Editing a Sensor System Definition File	8-3
8.1.3. Specifying a Sensor Signature	8-4
8.1.4. Adding a New Sensor Type	8-4
8.2. Probability and Selection Tables for Munitions	8-7
8.2.1. Hit Probability Tables	8-7
8.2.2. Damage Probability Tables	8-9
8.2.3. Detection Probability Tables	8-11
8.2.4. Ammo Select Tables	8-12
8.2.5. Configuring an Entity to Take Damage from a Munition	8-13
8.3. Configuring Formations	8-13
8.3.1. Formation Files	8-14
8.3.2. Creating a User-Defined Formation	8-15
8.4. Configuring Emitters	8-16
8.5. Configuring Radios	8-18
8.5.1. Configuring Transmitters	8-18
8.6. Configuring Joysticks	8-19
8.7. Configuring Obstruction Avoidance	8-20
8.7.1. Configuring Object Types	8-20
8.7.2. Configuring Vector Feature Types	8-21
8.8. Configuring Embarkation Ingress and Egress Points	8-22

Chapter 9 **Using VR-Forces with an External Communications Effects Server**

9.1. Introduction	9-2
9.2. Configuring VR-Forces	9-2
9.2.1. Enabling Radio Transmitter Publishing	9-3
9.2.2. Enabling the External Communications Model	9-3
9.2.3. Configuring the External Communications Model	9-4
9.3. Using QualNet	9-4
9.3.1. Running VR-Forces with QualNet	9-5

9.4. Example Scenario	9-5
9.4.1. Running the Scenario	9-6

Chapter 10 **Configuring 3D Front End Communication**

10.1. Communicating Information About the Eyepoint	10-2
10.1.1. Making the Eyepoint Visible to Other Applications	10-2
10.1.2. Configuring the Eyepoint Entity	10-3
10.1.3. Setting the DIS Protocol Version for Outgoing PDUs	10-5
10.1.4. Viewing and Attaching to Other Eyepoints	10-5
10.2. Controlling the Eyepoint from Other Applications	10-6
10.2.1. Enabling the Processing of View Control Messages	10-6
10.2.2. The Stealth Control Toolkit	10-6
10.2.3. Using Other MÄK Products to Control the Eyepoint	10-6
10.3. Using Mirroring	10-7
10.3.1. Sending View Control Messages Explicitly	10-7
10.3.2. Mirroring Display Options	10-8
10.3.3. Configuring the Frequency of View Control Messages	10-10
10.3.4. Configuring Multi-Channel Displays	10-11

Chapter 11 **3D Front End Configuration Files**

11.1. Specifying Parameters with the Startup Command	11-3
11.1.1. Setting Parameters from the Command Line	11-3
11.1.2. Setting Parameters by Loading a Configuration File	11-3
11.2. The <i>Stealth.mtl</i> File	11-4
11.3. The <i>params.mtl</i> File	11-5
11.4. Other Configuration Files	11-15
11.4.1. The <i>ModelMap.mtl</i> File	11-15
11.4.2. The Saved Views File	11-15
11.4.3. The <i>aggregateMap.mtl</i> File	11-15
11.4.4. Terrain Database Configuration Files	11-16
11.4.5. The <i>trailMap.mtl</i> File	11-16
11.4.6. The <i>stealthSoundMap.mtl</i> File	11-16
11.4.7. The <i>tacticalGraphicsMap.mtl</i> File	11-16
11.4.8. The <i>clusterRulesMap.mtl</i> File (XR mode Only)	11-16
11.4.9. The <i>infantryMap.mtl</i> File	11-16
11.4.10. The <i>rangeRingsMap.mtl</i> File	11-17
11.4.11. The <i>hazardCloudMap.mtl</i> File	11-17
11.4.12. The <i>guipreferences.mtl</i> File	11-17
11.5. Configuring Joysticks	11-17
11.6. Terrain Database Configuration Files	11-18
11.6.1. Specifying a Database's Origin	11-18
11.6.2. Multi-Part Databases	11-19
11.6.3. Specifying a Map Datum	11-19
11.6.4. Specifying the Default Eyepoint Position	11-20

11.7. Configuring TrueType Font-based Unit Icons (Windows Only) .	11-20
11.7.1. Unit Icon Parameters	11-21
11.7.2. Symbol Mapping File for TTF-based Unit Icons	11-22
11.8. Editing the ACF Configuration File	11-23

Chapter 12 Configuring 3D Entity Models

12.1. How the 3D Front End Maps Model Files to Entities	12-2
12.1.1. The model-map Command	12-2
12.1.2. Defining Symbolic Names for Entity Types and Models	12-3
12.2. Model Mapping Files Supplied with VR-Forces	12-4
12.2.1. The <i>ModelMap.mtl</i> File	12-4
12.2.2. The <i>aggregateMap.mtl</i> File	12-4
12.2.3. The <i>trailMap.mtl</i> File	12-4
12.2.4. The <i>modelset-2525B.mtl</i> Files (XR Mode Only)	12-6
12.3. Adding Your Own Model Mappings	12-6
12.3.1. Custom Model Mapping Files	12-6
12.4. Articulated Parts and Entity Appearances	12-7
12.5. Models Supplied with the 3D Front End	12-8

Chapter 13 Using the Model Mapper

13.1. Introduction to the Model Mapper	13-2
13.2. Starting the Model Mapper	13-2
13.2.1. The Model Mapper Interface	13-3
13.3. Mapping Entity Types to Model Files	13-6
13.3.1. Mapping Entity Types in the Entity Types Tab	13-7
13.3.2. Mapping an Entity Type to No Model	13-8
13.3.3. Changing a Model's Path Name	13-8
13.3.4. Specifying Textures for a Model	13-9
13.3.5. Removing a Mapping	13-9
13.3.6. Removing All Mappings	13-9
13.3.7. Saving Model Mappings	13-9
13.3.8. Opening Model Mapping Files	13-10
13.3.9. Entering Free Form Mappings	13-10
13.3.10. Deleting a Free Form Mapping	13-11
13.3.11. Deleting Unused Paths	13-11
13.4. Setting Model Mapper Preferences	13-11
13.4.1. Changing the Look and Feel of the Model Mapper	13-12
13.4.2. Specifying the Default Path Name	13-12
13.4.3. Specifying Absolute or Relative Model Paths	13-13
13.4.4. Specifying the 3D Preview Application	13-13
13.4.5. Turning Off the Overwrite Confirmation Prompt	13-13
13.4.6. Saving Preferences	13-13

Appendix A Object Parameters

A.1. The Structure of an Entity Parameter	A-2
A.2. The Parameter Type Hierarchy	A-3
A.2.1. Vrf Base Object Parameters	A-3
A.2.2. Vrf Object Parameters	A-7
A.2.3. Moving Object Parameters	A-8
A.2.4. Parameter Types for Entity Platforms	A-8
A.3. Entity Parameters	A-8
A.4. Object Type Enumerations	A-9
A.4.1. Object Super-type	A-9
A.4.2. AggregateKind	A-9
A.4.3. ObjectKind	A-10
A.4.4. Aggregate Category	A-10
A.4.5. Aggregate Specific	A-11
A.4.6. Point Object Category	A-12
A.4.7. LinearObjectKind	A-12
A.4.8. LinearObjectCategory	A-12
A.4.9. Interaction	A-12

Appendix B Editing the Create Menu

B.1. The Create Menu	B-2
B.2. The <i>vrf.ent</i> Configuration File	B-2
B.2.1. Object Blocks in <i>vrf.ent</i>	B-4
B.2.2. Reorganizing the Create Menu	B-4
B.2.3. Editing Menu Items	B-4
B.3. Adding New Entity Types to VR-Forces	B-5
B.3.1. Adding a New Entity or Control Object to <i>vrf.ent</i>	B-5
B.4. Editing the Overlay Object Menus	B-5

Appendix C Hazard Cloud Message Formats

C.1. Hazard Cloud Environmental Process Messages	C-2

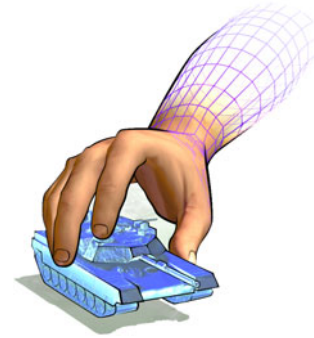
Appendix D The UTM Coordinate System

D.1. Standard Definition of the UTM	D-2
D.2. Arbitrary Offset UTM	D-2
D.3. Standard Offset UTM	D-3
D.4. Non-Offset UTM	D-3
D.5. USGS or Civilian Mapping UTM	D-3
D.6. Milgrid UTM	D-3
D.7. ModSAF UTM	D-3

Appendix E The WRM Specification

E.1. Background	E-2
E.2. General Information	E-3
E.3. Articulated Parts	E-4
E.4. Entity Appearances	E-5
E.4.1. Appearance Attributes Supported by 3D Front End Models ..	E-6
E.5. Attached Parts	E-8
E.5.1. Building Models that Can Have Attached Parts	E-8
E.5.2. Building Models that Can Be Attached Parts	E-8
E.6. Animation	E-9
E.7. WRM Model Examples	E-9

Index



Preface

This manual is written for persons who want to configure or customize VR-Forces®. The manual assumes that you are familiar with the use of VR-Forces, as described in VR-Forces User's Guide.

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

How This Manual is Organized

The chapters are organized as follows:

Chapter 1, *Optimizing Performance* describes how to monitor VR-Forces's performance in the Frame Rate Monitor and how to configure VR-Forces to improve performance.

Chapter 2, *VR-Forces Configuration Files* explains the syntax of configuration files and the parameters in the configuration files provided with VR-Forces.

Chapter 3, *Scenario Files* describes the files used to define a scenario and how to edit them.

Chapter 4, *Merging Scenarios* explains how to use the Scenario Merge tool to merge scenarios.

Chapter 5, *Editing Entity Models* explains how to use the Entity Editor to edit entity parameters, systems, and symbol mappings. It also explains how to create custom Create menus.

Chapter 6, *Advanced Entity Model Editing* explains how to use the OPD Editor to edit entity and system parameters. The OPD Editor lets you edit all entity parameters, while the Entity Editor only lets you edit some parameters.

Chapter 7, *The Object Parameter Database* describes the object parameter database, which contains parameters for entities and control objects, and how to edit it.

Chapter 8, *Configuring Object Behaviors* describes additional configuration files that VR-Forces uses to manage munitions, formations, and joystick use.

Chapter 9, *Using VR-Forces with an External Communications Effects Server* explains how to configure VR-Forces to work with external communications effects servers, such as QualNet.

Chapter 10, *Configuring 3D Front End Communication* explains how to configure the VR-Forces 3D Front End to broadcast its presence on the network and interact with other applications using view control messages.

Chapter 11, *3D Front End Configuration Files* describes the configuration files used for the 3D Front End.

Chapter 12, *Configuring 3D Entity Models* explains how to configure the entity models used in the 3D Front End.

Chapter 13, *Using the Model Mapper* explains how to use the Model Mapper tool to map OpenFlight models to entity types for display in the 3D Front End.

Appendix A, *Object Parameters* describes some of the parameters in the object parameter database.

Appendix B, *Editing the Create Menu* explains the format of the *vrf.ent* file and how to edit it to change the listings on the Create menu.

Appendix C, *Hazard Cloud Message Formats* lists details about hazard cloud formats used in the 3D Front End.

Appendix D, *The UTM Coordinate System* describes the different types of UTM coordinate systems.

Appendix E, *The WRM Specification* describes the specification used to display moving parts on models used in the 3D Front End.

VR-Forces Documentation

The VR-Forces documentation set is as follows:

- ♦ *VR-Forces User's Guide* describes how to install and use the VR-Forces program. It is distributed in printed format and as an online manual in Adobe Portable Document Format (PDF).
- ♦ *VR-Forces 3D Front End User's Guide* explains how to use the 3D features in the VR-Forces 3D Front End. Configuration of the 3D Front End is covered in *VR-Forces Configuration Guide*. It is distributed in printed format and in PDF format.
- ♦ *VR-Forces Configuration Guide* explains advanced features for configuring performance and how to edit VR-Forces configuration files. It includes documentation of the Entity Editor, OPD Editor, and Scenario Merge tools. It is distributed in printed format and in PDF format.
- ♦ *VR-Forces Developer's Guide* explains how to use the VR-Forces APIs. It is distributed in printed format and in PDF format.

- ♦ *VR-Forces Master Index* combines the index entries from the VR-Forces manuals to help you find references to subjects that are covered in two or more manuals or locate the manual in which a particular topic is discussed. It is distributed only in PDF format.
- ♦ *VR-Forces Migration Guide* collates API migration information for each release in the VR-Forces 3.x series. It is distributed only in PDF format.
- ♦ Online help. The VR-Forces GUI, the OPD Editor, and the TDB Tool have online help accessible from the Help menu.
- ♦ Class documentation. Classes are documented in HTML.
- ♦ *VR-Forces Release Notes*. Release documentation consists of release notes provided as a PDF file and in printed format.

The PDF files are in the *.doc* directory.

Online Manuals

The online manuals (PDF) use hypertext links for the entries in the table of contents, for all index entries, and for cross references. PDF documentation also supports full text searching.

To view and print PDF documents, you need a copy of the free Acrobat Reader software (version 5.0 or later). To obtain a copy, visit www.adobe.com.

MÄK Products

VR-Forces is a member of the MÄK Technologies® line of software products designed to streamline the process of developing and using networked simulated environments. The MÄK Technologies product line includes the following:

- ♦ VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA), the Distributed Interactive Simulation (DIS) protocol, and the Test and Training Enabling Architecture (TENA). VR-Link has built-in support for the 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.

- ♦ MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK 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 for the MÄK RTI *rtiexec*.

- ♦ VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a graphical user interface, that gives you a two dimensional view of a terrain database.

You can create and view local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities 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.

- ♦ MÄK Stealth and StealthXR. The Stealth displays a realistic, three-dimensional view of your virtual world. You can view this world from the inside of a simulated moving vehicle, or place the eyepoint at another moving or stationary location. The Stealth lets you switch rapidly among several predefined viewpoints while the simulation is underway.

StealthXR (exaggerated reality) is a separately-licensed component of the Stealth. It combines the features of traditional 2D and 3D displays into one exaggerated reality display. It provides a big picture understanding of a battlefield along with an immersive sense of perspective.

- ♦ MÄK 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 graphical user interface 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.
- ♦ MÄK Plan View Display. The Plan View Display (PVD) provides a two-dimensional, “big picture” of a virtual environment. It shows simulated entities, such as vehicles, soldiers, and tanks moving over a 2D-map display.
- ♦ 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 limited DIS translation.
- ♦ MÄK Gateway. The Gateway is a translator application that allows DIS exercises to operate as members of HLA federation executions that use the RPR FOM, thereby preserving the value of simulation code written to support DIS. The MÄK Gateway is a standalone application and does not require you to write additional code for its use.

How to Contact Us

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

Telephone

Call or fax us at:	Voice:	617-876-8085 (extension 3 for support)
	Fax:	617-876-9208

E-mail

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

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/licenses.php
Product version and platform information:	www.mak.com/support/productversions.php
For the free MÄK RTI trial version:	www.mak.com/products/rti.php
Support FAQ:	www.mak.com/support/faq.php

Post

Send postal correspondence to:	MÄK Technologies 68 Moulton St. Cambridge, MA, USA 02138

When requesting support, please tell us the product you are using, the version, and the platform on which you are running.

Document Conventions

This manual uses the following typographic conventions:

Monospaced	Indicates commands or values you enter.
Monospaced Bold	Indicates a key on the keyboard.
<i>Monospaced Italic</i>	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, (-h --help).
/	Indicates a directory. Since MÄK products run on both UNIX and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
<i>Italic</i>	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step 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.
i	Indicates supplemental or clarifying information.
!	Indicates additional information that you must observe to ensure the success of a procedure or other task.

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 *vrforces/doc* appears in the text as *./doc*.

Hardware and Operating System Naming Conventions

Unless otherwise specified, references to UNIX include Linux, regardless of the type of computer it is running on. References to PCs refer to Intel processor-compatible computers running a version of Microsoft Windows.

Mouse Button Naming Conventions

An instruction to click the mouse button, refers to clicking the primary mouse button, typically the left button for right-handed mice and the right button for left-handed mice. The popup 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

MÄK software products may use code from third parties. This section contains the license documentation required by these third parties.

Boost License

VR-Link, and all MÄK software that uses VR-Link uses some code which is distributed under the Boost License. All header files that contain Boost code are properly attributed. The boost website is: www.boost.org.

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

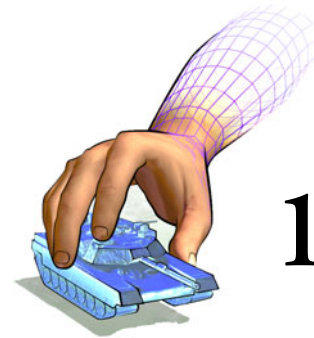
libXML and libICONV

VR-Link and all MÄK software which uses VR-Link links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MÄK Products. MÄK has made no modifications to these libraries.

- ♦ The LGPL license is available at: <http://www.gnu.org/licenses/lgpl.html>
- ♦ Information about IconV is at: <http://www.gnu.org/software/libiconv/>
- ♦ Information about LibXML is at: <http://xmlsoft.org/>

pThreads Library

VR-Exchange links with the pThreads win32 library. The library is distributed under the GNU Lesser General Public License (LGPL). MÄK has made no modification to this library. For information about the pThreads win32 library please see: <http://sourceware.org/pthreads-win32/>



Optimizing Performance

This chapter explains how you can view performance statistics for the front-end and back-end and how you can optimize GUI performance.

Introduction	1-3
Adjusting Refresh Rates	1-4
Changing the Frame Rate for the Display	1-4
Changing the Network Frame Rate	1-5
Monitoring Front-end and Back-end Frame Rates	1-5
Viewing Frame Rate Statistics.....	1-5
Configuring Statistics Reporting.....	1-7
Tuning GUI Performance	1-8
The Meeting Frame Rate Targets Level.....	1-8
The Not Meeting Frame Rate Targets Level.....	1-8
The Heavy Network Traffic Level.....	1-10
Miscellaneous Performance Configuration Options	1-12
Turn Off Track Histories	1-12
Turn Off the Display of Polygons	1-12
Limit Use of Object Transparency	1-13
Limit Use of Spot Reports	1-13
Configure Hazard Cloud Graphics	1-13
Configure Intervisibility Intersection Testing.....	1-13
Using Asynchronous I/O for DIS Exercises	1-13
Setting the Tick Rate	1-14
Specifying the Tick Rate for Components	1-14
Tuning the Network Interface	1-15
Tuning the State Repository Tick Rate	1-15
Setting Thresholds for Entity Updates	1-15

Optimizing Performance

[Tuning the VR-Forces Sleep Time](#) 1-17

[Configuring Spatial Subdivisions](#) 1-17

1.1. Introduction

This chapter describes ways that you can optimize performance through the GUI, the object parameter database, and command line options. You can also optimize performance programmatically. Please see Section 11.12, “[Optimizing GUI Performance](#)” in *VR-Forces Developer’s Guide* for details.

VR-Forces’s performance varies based on the following factors:

- ♦ Size of the terrain database
- ♦ Amount of memory available on your computer
- ♦ Computer speed and processing power
- ♦ The number of objects (entities and graphical objects) in the scenario and how active the entities are
- ♦ The number of objects you try to simulate on a given back-end
- ♦ The frequency of the tick rate
- ♦ The number of front-ends and back-ends you run on your computer.

VR-Forces supports the following methods for optimizing performance:

- ♦ Adjusting the refresh rates
- ♦ Turning off processor-intensive GUI features
- ♦ Adjusting the network processing time
- ♦ Specifying the tick rate for an entity or component
- ♦ Specifying a tick rate for the network interface
- ♦ Specifying the VR-Forces sleep time
- ♦ Specifying the number of objects that you expect to simulate in the exercise
- ♦ Optimizing the terrain. (For details, please see *MÄK Terrain Database Tool User’s Guide*.)

1.2. Adjusting Refresh Rates

VR-Forces supports two types of refresh rates, the network frame rate and the display frame rate. The network frame rate determines how many times per second the network is polled for new information. The display frame rate determines how many times per second the display is updated with information received from the network. You can set these frame rates on the General tab of the Display Options dialog box.

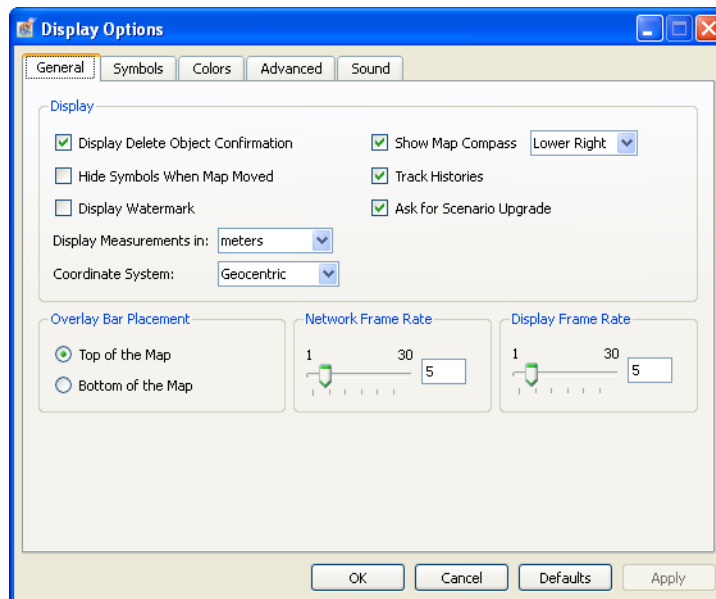


Figure 1-1. Display Options dialog box, General tab

1.2.1. Changing the Frame Rate for the Display

The display frame rate specifies the frequency with which VR-Forces updates the display. Updating the display less frequently may improve overall performance. However, depending on the speed at which entities are moving, it could introduce some jerkiness to their motion.

To change the display frame rate:

1. Choose Options → Display. The Display Options dialog box opens.
2. On the General tab (Figure 1-1), adjust the Display Frame Rate slider or enter a value from 1 to 30 in the text box.
3. Click OK.

The display frame rate does not affect the frame rate of the simulation source.

1.2.2. Changing the Network Frame Rate

The network frame rate specifies the frequency with which VR-Forces reads network input. Less frequent network reads may improve performance, but may result in lost information.

To change the network frame rate:

1. Choose Options → Display. The Display Options dialog box opens.
2. On the General tab ([Figure 1-1](#)), adjust the Network Frame Rate slider or enter a value from 1 to 30 in the text box.
3. Click OK.

1.3. Monitoring Front-end and Back-end Frame Rates

You can monitor the network and display frame rates in the Frame Rate Monitor. You can also monitor frame rate statistics for each back-end in your session, and you can specify:

- ♦ How frequently statistics are reported
- ♦ The sample size used to calculate statistics.

1.3.1. Viewing Frame Rate Statistics

The Statistics tab on the Frame Rate Monitor displays:

- ♦ The desired display and network frame rates, as specified in the Display Options dialog box (see [“Adjusting Refresh Rates,”](#) on page 1-4)
 - ♦ The actual frame rates
 - ♦ The GUI performance level
 - ♦ Frame rate statistics for each back-end in the session.
- To view frame rate statistics, choose Options → Frame Rate. The Frame Rate Monitor opens ([Figure 1-2](#)). The frame rate statistics are displayed in the window on the Statistics tab.

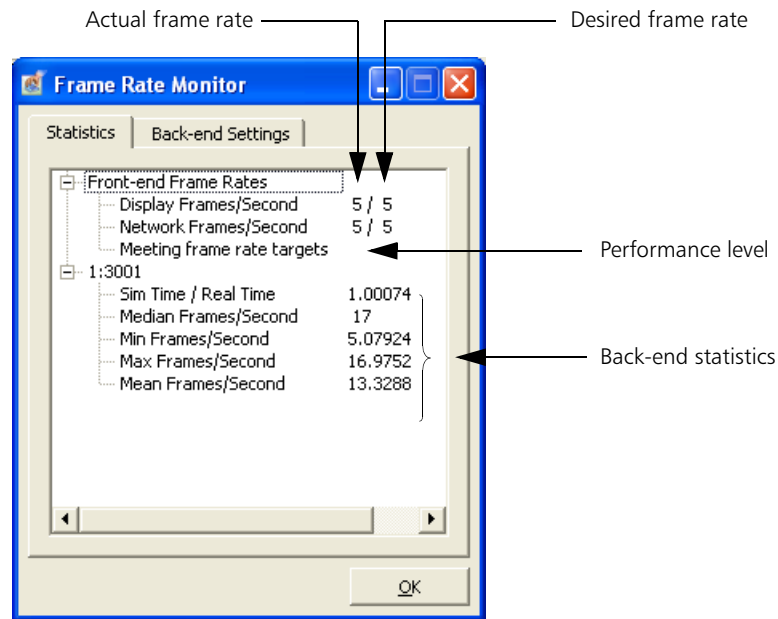


Figure 1-2. The Frame Rate Monitor

- To view the statistics for a back-end, click the plus sign next to its site:application number.

Table 1-1 describes the statistics displayed in the Frame Rate Monitor. For details about back-end statistics, please see [“Configuring Statistics Reporting,”](#) on page 1-7.

Table 1-1: Frame Rate Monitor statistics

Statistic	Description
Display Frames/Second	Actual frames displayed per second / the target display frame rate.
Network Frames/Second	Actual network frame rate / the target network frame rate.
Performance status	Indicates whether the GUI is meeting frame rate, not meeting the frame rate, or under heavy load.

1.3.2. Configuring Statistics Reporting

On the Back-end Settings tab of the Frame Rate Monitor, you can specify:

- How frequently a back-end reports frame rate statistics
- The sample size to use for frame rate statistics.

You can also configure logging of frame rate statistics with the `logFrameRateStatistics` parameter in `vrfSim.mtl`.

To specify the back-end statistics frequency and sampling size:

1. Choose Options → Frame Rate. The Frame Rate Monitor dialog box opens (Figure 1-2).
2. Select the Back-end Settings tab (Figure 1-3).

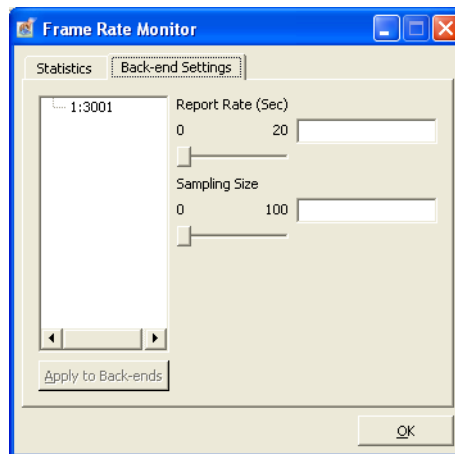


Figure 1-3. Frame Rate Monitor, Back-end Settings tab

3. In the list, select the back-end for which you want to configure the statistics display.
4. To specify the frequency, in seconds, that the back-end reports statistics, adjust the Report Rate slider.
5. To specify the number of frames that the back-end uses to calculate statistics, adjust the Sampling Size slider.
6. Click Apply to Back-ends.



When you apply changes to a back-end, the change affects reporting of statistics to all front-ends that are part of the session.

7. Repeat steps 3 through 6 for each back-end that you want to configure.

1.4. Tuning GUI Performance

The Frame Rate Monitor provides feedback about VR-Forces's performance ([Figure 1-2](#)). Performance is characterized as being at one of the following levels:

- ♦ Meeting frame rate targets
- ♦ Not meeting the frame rate targets
- ♦ Heavy traffic.

1.4.1. The Meeting Frame Rate Targets Level

When VR-Forces is meeting frame rate targets, it is polling the network and updating the display at the frame rates specified in the Display Options dialog box.

1.4.2. The Not Meeting Frame Rate Targets Level

When VR-Forces cannot meet the frame rate targets set in the Display Options dialog box, you can tune the GUI performance by specifying the percentage of the processing time of a frame to be used to process updates from the network and display them on the screen. To do this, you change the setting on the Display Options dialog box, Advanced tab. The value specified is a compromise between the goals of providing a smooth display of all entities in the map window and ensuring that no network information is lost.

If you specify a lower percentage, VR-Forces is more likely to provide a smooth display of entities, but it is also more likely to miss network updates, resulting in less accurate state information and the possibility of dropping entities from the display.

If you specify a higher percentage, the display of entity movement may become discontinuous, but it is less likely that VR-Forces will miss network updates, so the actual state information is likely to be more accurate.

To tune update processing:

1. Choose Options → Display. The Display Options dialog box opens.
2. Select the Advanced tab ([Figure 1-4](#)).
3. Adjust the network processing time slider.
4. Click OK.

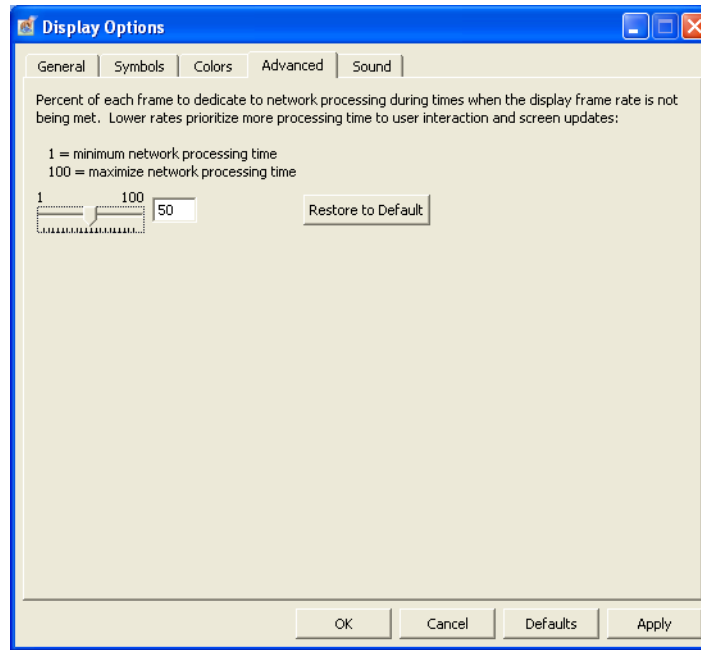


Figure 1-4. Display Options dialog box, Advanced tab

1.4.3. The Heavy Network Traffic Level

VR-Forces enters the heavy network traffic level when it determines that reading and processing information from the network is at a level that would make it difficult to use VR-Forces. The GUI may become less responsive to user input. In heavy network traffic level, VR-Forces reduces the frequency with which it polls the network for new information and displays it on the screen. This allows you to interact through the GUI. Heavy traffic mode is disabled by default, and, if enabled, should only happen rarely, if at all.

When VR-Forces is in heavy network traffic mode, it monitors its performance. If the performance improves, it enters the “not meeting target” or “meeting target” level. [Table 1-2](#) lists configuration parameters in *vrfGui.mtl* that support performance tuning.

Table 1-2: Performance tuning parameters

Parameter	Description
<code>consecutiveTimesOverThreshold</code>	Specifies the number of consecutive frames the system must exceed <code>networkTimeThreshold</code> before switching to heavy traffic mode. Enforced only if <code>tickManagingEnabled</code> is set to 1.
<code>networkTimeThreshold</code>	Specifies the maximum length of time, in milliseconds, that a network frame should take to execute. Enforced only if <code>tickManagingEnabled</code> is set to 1.
<code>networkFrameRateBuffer</code>	Specifies a value to use in determining whether or not to switch from “meeting targets” mode to “not meeting targets” mode. It multiplies <code>networkFrameRateBuffer</code> by the number of milliseconds the network frame took to execute. If the result is greater than the frame rate target, VR-Forces enters “not meeting targets” mode.
<code>tickManagingEnabled</code>	Enables (1) or disables (0) checking for heavy network traffic.

1.5. Optimizing VR-Forces 3D Front End Performance

VR-Forces 3D Front End has the following options for improving performance and scalability:

- ♦ You can specify the maximum number of entity models that are loaded at one time.
- ♦ You can specify ranges for visuals, detonation and fire effects, and sounds.
- ♦ You can filter HLA objects and interactions.
- ♦ You can disable computationally expensive features, as follows:
 - Set **updateHiddenTrajectories** to 0 (in *params.mtl*), to save processing time when track histories are disabled. The side effect of this setting is that histories are reset every time they are enabled.
 - Set **maxNumberOfEntityLabels** (in *params.mtl*). If the maximum number is less than the total number of entities, the labels closest to the eyepoint are drawn first.

1.5.1. Loading Entity Models and Special Effects

If VR-Forces loads a large enough number of entity models, it can run out of memory and crash. On Windows, processes usually have a 2 GB limit. On Linux, processes usually have a 4 GB limit. To avoid running out of memory, the terrain and loaded models must fit into the process's memory limits and the machine's RAM limits.

To stay below memory limits, VR-Forces can delay loading entity models until the entity is close to the eyepoint. To configure delayed loading, you can specify a maximum number of models and a display range. In addition to improving scalability, delayed model loading can improve performance. Only loaded models are updated for effects, trails, histories, and labels. The following parameters for configuring model and special effects loading are in *params.mtl*:

- ♦ **enableDelayedModelLoading** – Enables (1) or disables (0) delayed model loading.
- ♦ **maxNumberOfEntityModels** – Specifies the maximum number of loaded models. If more than N models fit within the desired range, the Nth closest is displayed.
- ♦ **maxLoadedEntityModelDistance** – Specifies the radius around the eyepoint, in meters, in which models are loaded.
- ♦ **fireDetonateVisualsRange** – Specifies the radius around the eyepoint, in meters, in which explosions and flak are displayed for detonate and fire interactions.

You can also configure the distance from the eyepoint within which sound effects are played. To configure the range for sound effects, edit the **soundRange** parameter in *params.mtl*.

1.5.2. Filtering HLA Traffic

You can filter the interaction and object classes that VR-Forces subscribes to. To filter out classes or interactions, uncomment the `filterFomClasses` and `filterFomInteractions` parameters in *params.mtl*. Add the classes and interactions that you want to filter out to the appropriate list.

1.6. Miscellaneous Performance Configuration Options

Disabling the following features may improve 2D front-end performance. Some are discussed in greater detail in the rest of this section.

- ♦ Hide the Minimap
- ♦ Turn off track histories
- ♦ Turn off spot reports
- ♦ Turn off the display of polygons
- ♦ Limit use of object transparency
- ♦ Do not display ellipses or bounding volumes for hazard clouds
- ♦ Reduce the types of intersections tested for intervisibility
- ♦ Use asynchronous I/O for DIS.

1.6.1. Turn Off Track Histories

If you turn off track histories, performance may improve.

- To turn off track histories, clear the Track Histories check box on the General tab of the Display Options dialog box, or set the `enableHistoryTracking` parameter in *vrfGui.mtl* to 0.

1.6.2. Turn Off the Display of Polygons

If you are using a terrain database that has a raster map and you do not need to view the underlying polygons, turning off the display of polygons can increase performance of panning and zooming, particularly when you are zoomed in on the database.

To turn off the display of polygons:

1. Choose Options → Terrain View. The Terrain View Options dialog box opens.
2. On the Drawing Mode tab, in the Terrain Drawing group box, clear the Polygons check box.
3. Click OK.

1.6.3. Limit Use of Object Transparency

Setting transparency for display objects, such as areas and range rings, can degrade performance.

1.6.4. Limit Use of Spot Reports

Use of spot reports can degrade performance. You may want to limit the number of entities for which they are generated or disable spot reports entirely.

1.6.5. Configure Hazard Cloud Graphics

If you are displaying hazard clouds, displaying images instead of ellipses or bounding volumes can dramatically decrease performance. Therefore, do not use images if performance is an issue.

1.6.6. Configure Intervisibility Intersection Testing

When it calculates intervisibility, VR-Forces can check for intersections of polygons, features, and entities. Depending on the complexity of the terrain database and the number of entities in a simulation, increasing the types of intersections for which VR-Forces tests can degrade performance. Therefore, if performance is an issue, minimize the intersection testing being done.

1.6.7. 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 entities may time out. You can mitigate this problem by using asynchronous I/O. To use asynchronous I/O, start with the `-I` command line option or set `useAsyncIO` to 1 in `vrfGui.mtl`.

1.7. 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 entity components are ticked, you make more CPU time available to other entities. For example you could decrease the tick rate for slow moving entities such as dismounted infantry entity or a command post, 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 an entity'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:

$$t_{\text{next}} = t_0 + \text{min-tick-period} +/\text{- rand}(\text{min-tick-period-variance}/2)$$

If you insert some sample values (in milliseconds), you get:

$$t_{\text{next}} = 100 + 50 +/\text{- rand}(50/2)$$

This means that the next tick will be no sooner than the current time plus 150 ms plus or minus a random value between 0 and 12.5 ms.

1.7.1. Specifying the Tick Rate for Components

In the object parameter database, you can specify a **min-tick-period** and **min-tick-period-variance** for each entity and for individual entity components. The component-level settings, if any, override the entity-level settings.

1.7.2. Tuning the Network Interface

You can tune the local and remote network interface subcomponents. Specify the `net-interface-min-tick-period` and `net-interface-min-tick-period-variance` parameters in the local and remote subcomponents blocks. If you reduce the rate at which the network interfaces are ticked, you can limit how often VR-Forces needs to do coordinate system conversion from network (geocentric) to database coordinates.



Reducing the frequency which which you tick network interface components could delay the discovery of objects in remote front-ends and back-ends.

1.7.3. Tuning the State Repository Tick Rate

You can tune the frequency with which state repositories get ticked. Specify the `state-repository-min-tick-period` and `state-repository-min-tick-period-variance` parameters in the local and remote subcomponents blocks. These parameters control how often VR-Forces recomputes the bounding volume of an object based on object geometry.

1.8. Setting Thresholds for Entity Updates

You can adjust the thresholds for entity movement and change in aggregate extents that determine when entity updates must be sent. As the number of entities and the level of entity activity increases, increasing the thresholds may aid performance, but may cause the movement of entities in VR-Forces and viewers such as the MÄK Stealth to appear less smooth.

The thresholds that you can set are:

- Translation Threshold – the amount of change in an entity’s position, in meters, that forces an update
- Rotation Threshold – the amount of change in an entity’s orientation, in radians, that forces an update
- Aggregate Dimensions – the percentage of change in an aggregate’s extents that forces an update.

By default, VR-Forces uses the default thresholds set by the VR-Link *DtThresholder*. (For details, please see Section 5.5.6, “[Setting Position and Orientation Thresholds](#)”, in *VR-Link Developer’s Guide*). You can set different default thresholds in *vrSim.mtl*. (For details, please see “[The vrSim.mtl Configuration File](#),” on page 2-14.)

When you change thresholds through the GUI, VR-Forces sends a message to all back-ends in the scenario to change their default thresholds for translation, rotation, and aggregate dimensions. The new thresholds remain in effect until the back-ends are closed. They apply to any scenario run during this time. Threshold changes made in the GUI override thresholds set in *vrSim.mtl*.

If your exercise sends periodic updates, such as a DIS heartbeat, the thresholds evaluate changes since the most recent heartbeat.

Changes made through the GUI have no effect in the following cases:

- If a back-end joins the scenario after the thresholds are sent, it uses the default VR-Link thresholds, or the values in the *vrfSim.mtl* file, if specified.
- If an object was instantiated with a customized *DtThresholder*.

Changes to the default thresholds do not get saved with VR-Forces settings. If you optimize thresholds and want to use the optimized thresholds in a future VR-Forces session, you must change the thresholds in *vrfSim.mtl* or apply them manually the next time you run VR-Forces.

To set entity update thresholds:

1. Choose Options → Networking Options. The Networking Options dialog box opens (Figure 1-5).

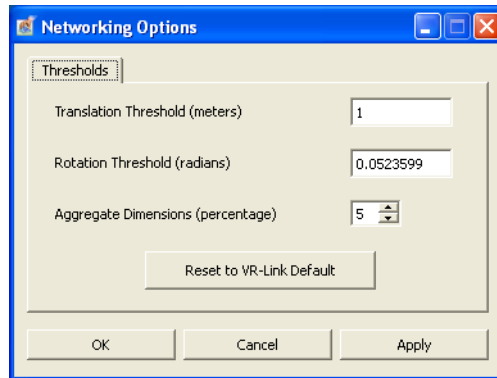


Figure 1-5. Networking Options dialog box

2. Change the values as desired or reset the thresholds to the default values set by VR-Link.
3. Click Apply to send the new thresholds to the back-ends.
4. Observe the effect of the new thresholds on the scenario. Make additional adjustments as desired.
5. When you achieve the optimal performance, record the thresholds that you used so that you can apply them the next time you run this scenario.
6. Click OK.

1.9. Tuning the VR-Forces Sleep Time

By default, the VR-Forces back-end sleeps for 10 milliseconds each frame, to give other applications more access to the CPU. However, when you are simulating a larger number of entities, this 10 ms can start to become a significant fraction of the current frame time (at 20 Hz, VR-Forces is now sleeping 200 ms out of each second, or 20%). The `-e` command line option allows you to change the sleep time, for example:

```
vrfSim -e 0.0
```

If you are simulating a large number of entities, in a Windows environment, we recommend setting the sleep time to 0.0. The process scheduler will still give CPU time to other processes, including the GUI, but if no other processes are demanding CPU, then VR-Forces will get nearly all of the processor time. If GUI performance becomes unacceptable, you will have to increase the sleep time or run the front-end on a different computer.

1.10. Configuring Spatial Subdivisions

Spatial subdivisions are a tuning parameter for objects. You can adjust the spatial subdivision of objects during runtime. Object spatial subdivisions are calculated every tick. The default settings for spatial subdivisions are appropriate for most simulations. However, fine tuning them may benefit scenarios that have large numbers of objects, particularly if they are spread out.



Setting the spatial subdivision settings through the GUI may be computationally expensive, and doing so during runtime may affect performance, because the simulation waits until the subdivisions are reconfigured. It is best to do this while the simulation is paused.

The settings are not saved with the scenario, so they must be reset each time you load the terrain.

To specify spatial subdivisions:

1. Choose Options → Configure Spatial Subdivision. The Configure Spatial Subdivisions dialog box opens (Figure 1-6).

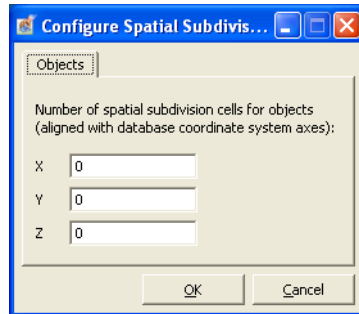
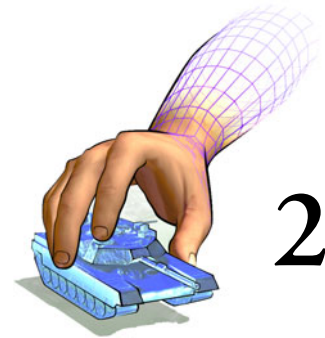


Figure 1-6. Configure Spatial Subdivision dialog box

2. Enter X, Y, and Z values for objects on the Objects tab.
3. Click OK.



VR-Forces Configuration Files

This chapter describes the syntax of VR-Forces configuration (MTL) files and introduces the configuration files used by VR-Forces.

MÄK Technologies Lisp (MTL) Files	2-2
Using Environment Variables in an MTL File	2-2
Specifying Lists of Lists	2-3
Using Conditional Statements	2-3
Configuration File Descriptions	2-4
The vrfGui.mtl Configuration File	2-5
The vrfSim.mtl Configuration File	2-14
Symbol Mapping Files	2-19
The plugin.mtl File	2-21
The commonObjects.mtl File	2-21
The soundmap.mtl File	2-21
The rangeRings.mtl Configuration File	2-21
The boundingVolume.mtl Configuration File	2-22
Configuring Overlay Objects	2-22
Configuring Communication Models	2-23
Simple Radio Communication Model	2-23
External Communication Model	2-23
Custom Communication Models	2-24

2.1. MÄK Technologies Lisp (MTL) Files

Many of the configuration files for MÄK Technologies products are ASCII text files that use MÄK 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 EnableSound 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 MÄK 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 (from the MÄK Stealth), 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 to the Stealth. 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/Vapor-Trail" "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")
```

2.1.1. Using Environment Variables in an MTL File

If you want to use an environment variable in an MTL file, use the following syntax:

```
(setqb parameter_name (getenv (quote env_var)))
```

For example:

```
(setqb disDestAddr (getenv (quote DEST_ADDR)))
```

2.1.2. 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))
```

2.1.3. 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)
  )
)
```

2.2. Configuration File Descriptions

VR-Forces (*vrfSim* and *vrfGui*) uses the following configuration files. Unless otherwise noted, they are in *./config*. Configuration files used by the 3D Front End are described in Chapter 11, *3D Front End Configuration Files*.

- ♦ *aggregates.mtl* specifies the list of aggregates for the Aggregate As dialog box.
- ♦ *appearance.mtl* specifies the appearance attributes that can be set in the Set Appearance dialog box.
- ♦ *boundingVolume.mtl* specifies bounding volumes of entities for use when calculating intervisibility intersections.
- ♦ *capabilities.mtl* specifies capabilities that can be set in the Set Capabilities dialog box.
- ♦ *commonObjects.mtl* configures various graphical objects that do not fit logically into the other configuration files.
- ♦ *componentCodeGeneratorConfig.mtl* configures the VR-Forces Component Code Generator application.
- ♦ *entityEditorUi.mtl* configures the Entity Editor.
- ♦ *overlayMenu.mtl* and *overlayMenu-bmp.mtl* specify the overlay objects listed in the overlay objects menus and the Overlay Palette.
- ♦ *overlayPalette.mtl* and *overlayPalette-bmp.mtl* (in *./data/simulationModelSets/<modelset>/gui*) configure how overlay objects are drawn and which symbols are used for them.
- ♦ *plugin.mtl* configures which plug-ins get loaded, if any.
- ♦ *rangeRings.mtl* configures the size and style of range rings.
- ♦ *soundmap.mtl* configures the sound files used to represent entities.
- ♦ *symbolmap-bmp.mtl* (in *./data/simulationModelSets/<modelset>/gui*) maps entity types to bitmap files.
- ♦ *symbolmap-images.mtl* and *symbolmap-images-perspective.mtl* (in *./data/simulationModelSets/<modelset>/gui*) map entity types to representational images.
- ♦ *symbolmap-ttf.mtl* (in *./data/simulationModelSets/<modelset>/gui*) maps entity types to TrueType fonts.
- ♦ *vrf.ent* specifies the entities, aggregates, and control objects listed on the Create menu if you use a custom Create menu. For information about this file, please see [“The vrf.ent Configuration File,”](#) on page B-2.
- ♦ *vrfGui.mtl* specifies generic configuration information and protocol-specific configuration values for the front-end.
- ♦ *vrfGuiSettings.mtl* saves display settings for the front-end.
- ♦ *vrfSim.mtl* specifies configuration information for the back-end.

If you want to use an alternative GUI configuration file, you can replace the default *vrfGui.mtl* file or you can start VR-Forces with the `-l` option and specify a different file, for example:

```
vrfGui -l "../config/myconfigfile.mtl"
```

2.2.1. The *vrfGui.mtl* Configuration File

The *vrfGui.mtl* file configures parameters for the 2D front-end. You can override the configuration file using command-line options.

[Table 2-1](#) describes the parameters and lists the equivalent command-line option, where applicable.

Table 2-1: *vrfGui.mtl* parameters (Sheet 1 of 10)

Parameter	Description	Option
<code>addSymbolMapFile</code>	Specifies a symbol map file to use. You can have multiple instances of this parameter.	<code>-m</code>
<code>aggregatesFilename</code>	Specifies the aggregates configuration file to load for the Aggregate As dialog box. Default: <i>aggregates.mtl</i> .	
<code>appearanceFile</code>	Specifies the configuration file for entity appearance attributes.	
<code>appNumber</code>	Specifies the application number.	<code>-a</code>
<code>askSaveScenarioIfChanged</code>	Specifies whether (1) or not (0) to prompt a user to save a scenario when they run it for the first time or when they close it.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 2 of 10)

Parameter	Description	Option
availableTaskSetRetrievalScheme	Specifies how context sensitivity is implemented for the Task and Set menus. The options are: 0: Retrieve tasks and sets by name 1: Retrieve tasks and sets by entity type 2: Do not retrieve tasks and sets Default: 0. This option is designed to support exercises in which multiple instances of VR-Forces have different configurations of plug-ins that support different task and set commands. For example the B-HAVE Module for VR-Forces adds commands to the Task and Set menus. If some entities in a simulation are simulated by VR-Forces that is running B-HAVE and some are not, the lists of available tasks and sets returned from the back-end would be incorrect for some entities if they are treated the same way. If you set this parameter to 0, the front-end sends a separate request for each entity, using its name. In this case the context sensitive list is always correct. If you have a large number of entities, you can reduce the number of requests by using entity type (option 1). However, the list may be incorrect for some entities.	
backendApplicationName	Specifies the name of the back-end executable. Default: <i>vrfsim.exe</i> .	
backendAppNum	Specifies the default application number for the back-end. Default: 3001.	
backendSiteId	Specifies the default site ID for the back-end. Default: 1.	
backendTimeoutInterval	The amount of time that the front-end keeps a back-end in its list of back-ends without getting a status message from the back-end.	
batchScenarioFilename	Specifies the scenario to load when VR-Forces starts. If <i>scenarioFilename</i> and <i>batchScenarioFilename</i> are both specified, the batch scenario is loaded.	
boundingVolumesFile	Specifies the path of the <i>boundingVolume.mtl</i> file.	-b
capabilitiesFile	Specifies the configuration file for capabilities.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 3 of 10)

Parameter	Description	Option
commandLineDiscoveryTimeout	Specifies, in milliseconds, how long the front-end waits to discover a back-end when it loads a scenario from the command line.	
defaultFreehandSamplingRate	Specifies the number of points to sample when drawing a freehand line.	
defaultOverlayMenu	Specifies the name of the <i>overlayMenu.mtl</i> file to use.	
defaultScenario	Specifies the location of the <i>default.scn</i> file.	
defaultSettingsFile	Specifies the name of the settings file to load and save settings to.	-d
defaultSymbolMapFile	Specifies the symbol map file for entity icons.	
defaultTerrainDatabasePath	Specifies the starting directory for the Open dialog box for loading terrain databases.	
doNotUseSettingsFile	Specifies whether to load or save settings (0) or not to load or save settings (1).	-D
DtNotifyLevel	Specifies the notification level for application level console messages.	
enableEmitterHandling	Not implemented. Do not edit this parameter.	
enableLogoOverlay	Enables (1) or disables (0) display of a logo overlay in the map area.	
enableSoundSystem	Enables (1) or disables (0) sound effects.	
hostAddrString	Specifies an ethernet address to receive files from the server during a save. This is for use on computers that have multiple network cards and on which you do not want to use the first network card for this purpose.	-H
language	Specifies the language to use in the VR-Forces GUI.	-G
makePathsRelative	Specifies that pathnames in the scenario file be saved using absolute paths (0) or relative paths (1). Default: 1.	
maxDrainInputTime	Sets the maximum time to wait in drain-Input() before returning. For the DMSO and MÄK RTIs, the default values will drain the entire input stream before returning. For the Pitch RTI, the default values read a single input from the input stream, so set to a value other than -1 to process more information. Use with DIS or HLA.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 4 of 10)

Parameter	Description	Option
maximumSignalTimeDifference	Specifies the maximum number of seconds between two signals on a transmitter frequency within which to assume that the second signal is responding to the first signal.	
objectConsoleHistoryLimit	Specifies the maximum number of console messages to be cached for each object.	
objectInformationToolbar	Enables (1) or disables (0) display of the Object Information Toolbar.	
overlayLogoFile	Specifies the image to display if the overlay logo is enabled.	
overlayLogoPosition	Specifies the position of the overlay logo on the map. Position is a value from 1 through 9 based on the following grid: 7 8 9 4 5 6 1 2 3 Default: 7.	
overlayLogoPositionX	Specifies the logo position by X and Y pixel coordinates. If not -1, overrides the overlay-LogoPosition parameter.	
overlayLogoPositionY		
overlayLogoScaleFactor	The amount to scale the logo image. Default: 1.	
patternsFile	Specifies the entity type patterns file to load for the Pattern tab on the conditional expression dialog boxes.	
pluginsDirectory	Specifies the directory in which configuration (MTL) files for plug-ins are located. The directory must end with a slash, for example, <i>../plugins/</i> .	
preserveEmptyOverlayLayers	Specifies whether (1) or not (0) to keep an overlay from which all objects have been deleted. Default: 0 (remove empty layers).	
processSignalInteractions	Enables (1) or disables (0) processing of transmitter information. Default: 1.	
qtAssistantLocation	Specifies the location of the Qt Assistant executable.	
qtHelpFile	Specifies the location of the help file for the Qt Assistant browser. This setting overrides the setting in <i>vrfGui.mtl</i> .	
qtHelpFile	Specifies the location of the online help table of contents file. The <i>qtHelpFile</i> parameter in <i>vrfGui.mtl</i> overrides this parameter.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 5 of 10)

Parameter	Description	Option
rangeRingsFile	Specifies the configuration file for range rings.	
remoteControllerReceivePort	Specifies the port to be used to receive scenario files from a back-end in response to a save scenario request.	
saveTimeoutInterval	The amount of time the front-end waits for a scenario to be saved before it resends a save message. The default is 60 seconds, so if you know that your scenario takes longer than this to save, you must increase this value.	
scenarioFileName	Specifies the scenario to load when VR-Forces starts. If scenarioFilename and batchScenarioFilename are both specified, the batch scenario is loaded.	
scenarioTimeoutInterval	The amount of time that the front-end waits for a scenario to load before it sends another load scenario message. The default is 60 seconds, so if you know that your scenario takes longer than this to load, you must increase this value.	
separateObjectLists	Enables (1) or disables (0) use of separate toolbars for the various object selection lists. Default: 0 (Objects tab has all object lists.)	
sessionId	Specifies the session ID to use.	-i
ShowGroundTruthForForce	Specifies the forces for which you want VR-Forces to display ground truth.	
showSplashScreen	Enables (1) or disables (0) display of the splash screen at startup.	
ShowSpotReportsForForce	Specifies the forces for which you want to display spot reports.	
simulationModelSet	Specifies the default simulation model set or sets. If specifying multiple SMSs, separate them by a semi-colon.	
siteld	The site ID.	-s
SpotReportSystemEnabled	Enables (1) or disables (0) VR-Forces' ability to display spot reports.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 6 of 10)

Parameter	Description	Option
storeLineValuesMode	Specifies whether or not to preserve line creation settings for color, width, sampling and force settings. Possible settings are: 0 – Do not store any values 1 – Store for freehand lines only 2 – Store for all lines. This only applies to lines and routes and is only valid for the current session.	
terrainDatabase	Specifies a default terrain database to load at startup. If you load a database at startup, you cannot create a scenario with it.	
terrainDatabase	Specifies the terrain database to load by default at startup.	
timeMultiplierLowerBound	Specifies the lower boundary of the range of the Time Multiplier toolbar. Default: 0.066 (1/15).	
timeMultiplierUpperBound	Specifies the upper boundary of the range of the Time Multiplier toolbar. Default: 15.	
useConsole	Enables (1) or disables (0) display of the console window.	-C
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. Please see the comments in the file.	
usingGlobalMiniMap	Specifies whether to use (1) or not to use (0) a global Minimap.	-g
viewControlMessages	Enables (1) or disables (0) response to view control messages.	
Remapping Parameters	For details about remapping, please see Section 3.2.5, “ Working with Multiple Back-ends ” in <i>VR-Forces User’s Guide</i> .	
autoRemapInCombinedMode	Enables (1) or disables (0) automatic remapping of objects when you run a scenario in combined mode. VR-Forces remaps objects to the back-end specified by the <code>backendSiteld</code> and <code>backendAppNum</code> parameters (default: 1:3001). Default: 1.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 7 of 10)

Parameter	Description	Option
<code>autoRemapInSeparateMode</code>	Enables (1) or disables (0) automatic remapping of objects when you run a scenario in separate mode. Objects are mapped to the back-end specified by the <code>separateModeRemapSiteId</code> and <code>separateModeRemapAppNum</code> parameters. Default: 0.	
<code>separateModeRemapSiteId</code>	Specifies the site ID of the back-end that you want to remap objects to if you set <code>autoRemapInSeparateMode</code> to 1. Default: 1.	
<code>separateModeRemapAppNum</code>	Specifies the application number of the back-end that you want to remap objects to if you set <code>autoRemapInSeparateMode</code> to 1. Default: 3001.	
<code>batchModeRemapSiteId</code>	Specifies the site ID of the back-end that you want to remap objects to in batch mode. Default: 1.	
Trajectory History Parameters		
<code>enableHistoryTracking</code>	Enables (1) or disables (0) recording of trajectory histories. Disabling trajectory histories may improve performance. You cannot override this setting in the GUI. Default: 1.	
<code>decayTrackHistory</code>	Specifies that an entity's trajectory history decays even if it is not moving (1) or does not decay when it is not moving (0). Setting this parameter to 1 may affect performance. Default: 0.	
<code>minTrackHistoryMovement</code>	Specifies the minimum distance, in meters, between points used to create a trajectory history. Default: 60.	
<code>trackHistoryTimeout</code>	The length of time (simulation seconds) to keep any point in the trajectory history list. Set to -1 to never timeout the points on the line. Default: 60.	
Performance Tuning Parameters		
<code>consecutiveOverFrameRateThreshold</code>	Specifies the number of consecutive frames the system must exceed <code>networkTimeThreshold</code> before switching to heavy network traffic mode. Enforced only if <code>tickManagingEnabled</code> is set to 1.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 8 of 10)

Parameter	Description	Option
hideOffScreenSymbols	Specifies whether or not to hide symbols when they go off screen instead of destroying them. Setting this parameter to 1 may improve performance for large exercises.	
networkTimeThreshold	Specifies the maximum length of time, in milliseconds, that a network frame should take to execute. Enforced only if tickManagingEnabled is set to 1.	
networkFrameRateBuffer	Specifies a value to use in determining whether or not to switch from “meets frame rate target” mode to “not meeting target” mode. Multiply networkFrameRateBuffer by the number of milliseconds the network frame took to execute. If the result is greater than the frame rate target, enter “not meeting target” mode.	
slowNetworkFrameOverride	Specifies the number of milliseconds to wait between network frames before polling the network again. Enforced only if tickManagingEnabled is set to 1. Default: Twice the length of the longest network frame or 5 seconds, whichever is less.	
tickManagingEnabled	Enables (1) or disables (0) checking for heavy network traffic.	-e
batchModeRemapAppNum	Specifies the application number of the back-end that you want to remap objects to in batch mode. Default: 3001.	
HLA Parameters	For details, please see Section 4.7.3, “ Command Line Options for HLA Federations ” In <i>VR-Forces User’s Guide</i> .	
destroyFedExec	Specifies whether (1) or not (0) the <i>DtExerciseConn</i> destructor should try to destroy the federation execution. Default: 1.	
execName	The federation execution name.	-x
federateName	The federate name. The federateName parameter in <i>vrfGui.mtl</i> overrides this parameter.	-N
fedFileName	Specifies the FED file name.	-F
fomMapperLib	A FOM Mapper library.	-f
fomMapperInitData	An initialization string. --fomMapperInitData	

Table 2-1: *vrfGui.mtl* parameters (Sheet 9 of 10)

Parameter	Description	Option
lookahead	Specifies the lookahead value if time regulating mode is enabled.	
minDrainInputTime	The minimum time to stay in drainInput() before returning.	
rprFomVersion	The RPR FOM version. --rprFomVersion	
runInTimeConstrainedMode	Enables (1) or disables (0) time constrained mode for HLA time management.	
runInTimeRegulatingMode	Enables (1) or disables (0) time regulating mode for HLA time management.	
sendFedTime	Enables (1) or disables (0) sending and receipt of interactions in time stamp order. Default 0.	
simTimeScale	Specifies the desired ratio of simulation time to real time if time constrained mode is enabled.	
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.	
useAdvisories	Enables (1) or disables (0) HLA advisory messages.	
DIS Parameters	For details, please see Section 4.7.4, " Command Line Options for DIS Exercises " in <i>VR-Forces User's Guide</i> .	
addMulticastAddr	The multicast address.	-S
appldRange	Specifies the application ID range to use to identify entities 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 mode.	-A
disPort	The port number.	-P
exerciseld	The exercise ID.	-x
maxDrainInputReads	The maximum number of reads to make in drain input before returning	
mcastTtl	Specifies the multicast time to live.	

Table 2-1: *vrfGui.mtl* parameters (Sheet 10 of 10)

Parameter	Description	Option
	--mcastTtl	
recvBufferSize	Specifies the receive buffer size, as a positive value. A value of -1 means use the system default.	
	--recvBufferSize	
sendBufferSize	Specifies the send buffer size, as a positive value. A value of -1 means use the system default.	
	--sendBufferSize	
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.	
useAsyncIO	Enables (1) or disables (0) use of asynchronous I/O, which can help reduce dropped packets under heavy network load. Default: 0.	-I

2.2.2. The *vrfSim.mtl* Configuration File

The *vrfSim.mtl* file configures back-ends.

Table 2-2: *vrfSim.mtl* parameters

Parameter	Description	Option
appNumber	Specifies the application number. Default: 3001.	-a -p
batchFile	Specifies a batch file to run.	-B
cgfDispatcherReceivePort	Specifies the port to be used by the <i>DtCgfDispatcher</i> 's file transporter to receive order of battle and plan files from the <i>DtVrfRemoteController</i> .	
daemon	Runs vrfSim as a daemon process (Linux only.) vrfSim will not attempt to read characters from a console, and will fork a background process on startup. It will exit cleanly on receipt of the SIGTERM, SIGQUIT, SIGINT signals. Default: Off.	
	--daemon	
defaultMainLoopSleepInterval	Sets the sleep interval for DtCgf::tick(). By adjusting how long your application sleeps before it ticks the <i>DtCgf</i> , you can tune the performance of the main application loop.	

Table 2-2: *vrfSim.mtl* parameters

Parameter	Description	Option
defaultParameterDatabasePath	Specifies the location to search for the object parameter database if the full path is not specified in a scenario file.	
defaultTerrainDatabasePath	Specifies the location to search for the terrain database if the full path is not specified in a scenario file.	
doNotUseConsole	Disables (1) or enables (0) writing of output to the console. Default: 0.	
DtNotifyLevel	Sets the notification level (0-4) for information and warning messages. Default: 2.	-n
entVrfDataTimeThreshold	Specifies how often, in seconds, VR-Forces entity data is sent. If you specify -1.0, the update is sent only if some item in the vrfDataObject has changed.	
envVrfDataTimeThreshold	Specifies how often, in seconds, VR-Forces environmental data is sent. If you specify -1.0, the update is sent only if some item in the vrfDataObject has changed.	
exerciseStartTime	Specifies the exercise time at the start of the simulation.	-k
forceParameterDbReload	If set to 1 (default), forces the back-end 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 OPD as the previous one (for example, in a rewind), the OPD is not reloaded.	
ignoreAdvisories	Turns off HLA advisory messages.	
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 1.3, HLA 1516.) 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.	

Table 2-2: *vrfSim.mtl* parameters

Parameter	Description	Option
maxDrainInputTime	Sets the maximum time to wait in drainInput() before returning. For the DMSO and MÄK RTIs, 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.	
pluginsDirectory	Specifies the directory in which configuration (MTL) files for plug-ins are located. The directory must end with a slash, for example, <i>../plugins/</i> .	
publishSimulationTime-ForStealth	Specifies whether or not to publish the simulation time to the network so that the MÄK Stealth can set its own simulation clock to work in conjunction with the simulation time.	
scenarioFileName	Specifies the scenario file to load.	-I
sessionId	Specifies the session ID.	-i
simMgmtPduProcessLevel	Specifies how simulation management PDUs are processed: 1 – Processes PDUs sent to a specific back-end. Wildcards are ignored. 2 – Processes only wildcarded PDUs. Ignores those sent to specific back-ends. 3 – Processes all PDUs.	
simTime	Specifies the starting simulation time.	-j
simulationModelSet	Specifies the default simulation model set or sets. If specifying multiple SMSs, separate them by a semi-colon.	
siteld	Specifies the site ID. 0 = use internal default.	-s -t
sleepInterval	Specifies how much time, in seconds, the <i>DtCgf</i> class should sleep between calls to tick(). Default: 0.01.	-e
startInRunMode	Specifies that the simulation start in run mode (1), rather than pause mode (0). Default: 0.	
statusUpdateHeartbeat	Specifies how often, in seconds, the back-end status heartbeat is sent.	
terrainFileName	Specifies the terrain database to load. This parameter is overridden by the scenario file if one is specified in scenarioFileName.	-T

Table 2-2: *vrfSim.mtl* parameters

Parameter	Description	Option
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, please see the comments in the file.	
HLA Parameters	For details, please see Section 4.7.3, “Command Line Options for HLA Federations” in <i>VR-Forces User’s Guide</i> .	
destroyFedExec	Specifies whether (1) or not (0) the <i>DtExerciseConn</i> destructor should try to destroy the federation execution. Default: 1.	
fomMapperLib	A FOM Mapper library.	-f
fomMapperInitData	An initialization string. --fomMapperInitData	
execName	The federation execution name.	-x
federateName	The federate name.	-N
fedFileName	Specifies the FED file name.	-F
minDrainInputTime	The minimum time to stay in drainInput() before returning.	
rprFomVersion	The RPR FOM version. (0.5, 0.8, 1.0, 2.0) --rprFomVersion	
runInTimeManagementMode	Enables (1) or disables (0) use of time management.	
sendFedTime	Enables (1) or disables (0) sending and receipt of interactions in time stamp order. Default 0.	
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.	
useAdvisories	Enables (1) or disables (0) HLA advisory messages.	
DIS Parameters	Please see “Command Line Options for DIS Exercises,” on page 4-30 in <i>VR-Forces User’s Guide</i> .	
addMulticastAddr	The multicast address.	-S
appldRange	Specifies the application ID range to use to identify entities 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.	

Table 2-2: *vrfSim.mtl* parameters

Parameter	Description	Option
defaultObjectTimeout	Specifies the DIS object timeout threshold, in seconds. If VR-Forces does not receive a heart-beat for an object for the time period specified, it removes the object from the exercise.	
destAddrString	Destination address for point-to-point communications mode.	-A
disPort	The port number.	-P
exerciseld	The exercise ID.	-x
maxDrainInputReads	The maximum number of reads to make in drain input before returning	
mcastTtl	Specifies the multicast time to live. --mcastTtl	
recvBufferSize	Specifies the receive buffer size, as a positive value. A value of -1 means use the system default. --recvBufferSize	
sendBufferSize	Specifies the send buffer size, as a positive value. A value of -1 means use the system default. --sendBufferSize	
useAbsoluteTimeStamps	If set to 1, use absolute time stamps. If set to 0, use relative time stamps. Default: 0.	
useAsyncIO	Enables (1) or disables (0) use of asynchronous I/O, which can help reduce dropped packets under heavy network load. Default: 0.	-I

2.2.3. Symbol Mapping Files

VR-Forces loads a symbol map file to map entity types to symbols for display purposes. Symbol map files are in `./data/simulationModelSets/<modelset>/gui`. The default symbol map file is specified in `vrfGui.mtl`.

If you want to map entities to different symbols, you have the following options:

- ♦ Edit the `addSymbolMapFile` parameter in `vrfGui.mtl`.
- ♦ Start VR-Forces with the `-m` option and specify a symbol map file.
- ♦ Change the symbol map file after you start VR-Forces.

Each entry in a symbol map file has the format:

```
(symbol-map (list object_enumeration) "file_type" "mapping")
```

where:

- ♦ `object_enumeration` is an enumeration that identifies the entity type of the object (see [Figure 2-1](#)).
- ♦ `file_type` identifies the type of graphic (bmp, ttf, uttf)
- ♦ `mapping` identifies the graphic file or font mapping for the object.

For example:

```
(symbol-map (list 1 -1 1 1 -1 -1 -1 -1 -1) "bmp"
  "../data/symbols/bmp/groundF.bmp")
```

If you want to add additional mappings to the configuration files, add new symbol-map entries.

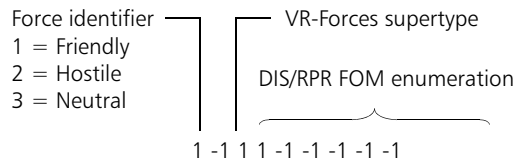


Figure 2-1. Symbol map enumeration syntax

Symbol Mapping Syntax for Animated Images

VR-Forces supports animated images for entities and effects such as explosions. The animated images are a sequence of raster images that are displayed rapidly to produce the effect of motion. The syntax for an animated image is as follows:

```
(symbol-map (list object_enumeration) "file_type" (list "mapping1" ...  
"mappingn") frame_interval loop)
```

where:

- ♦ *object_enumeration* is an enumeration that identifies the entity type of the object (see Figure 2-1). It includes the supertype value used in VR-Forces.
- ♦ *file_type* identifies the type of graphic (anim-bmp).
- ♦ *mapping* identifies the graphic files for the animation.
- ♦ *frame_interval* specifies the amount of time between each frame, in seconds.
- ♦ *loop* specifies whether or not to loop the animation. 1 = loop; 0 = do not loop.

For example:

```
(symbol-map (list 1 -1 1 2 225 4 1 -1 -1) "anim-bmp"  
(list "../data/symbols/images/C-130_Hercules0000.png"  
"../data/symbols/images/C-130_Hercules0001.png"  
"../data/symbols/images/C-130_Hercules0002.png") 0.1 1)
```

Symbol Mapping Syntax for TrueType and Unicode TrueType Fonts

The mapping portion of a symbol map entry for a TrueType font (ttf) is as follows:

```
"ttf" "font_name" "character"
```

For example, the mapping for a generic tracked entity is:

```
(symbol-map (list 0 -1 1 1 -1 1 -1 -1 -1) "ttf" "MapSym NK Land" "A")
```

The mapping portion of a symbol map entry for a unicode TrueType font (uttf) is as follows:

```
"uttf" "font_file" "display_character" "outline_character"  
"fill_character"
```

For example, the mapping for a generic tracked entity is:

```
(symbol-map (list 0 -1 1 1 -1 1 -1 -1 -1) "uttf" "MilStd 2525b" "0000"  
"4929" "0025")
```

You must have a viewer that can view unicode fonts to see which characters to use.

Changing the Symbol Map File During Runtime

You can change the symbol map file being used while VR-Forces is running.

To load a symbol map file:

1. Choose Options → Display. The Display Options dialog box opens.
2. Select the Symbols tab.
3. In the Symbol Mapping group box, select the top option button.
4. In the drop-down list, select the symbol set you want to use.
5. Click OK.

2.2.4. The *plugin.mtl* File

The *plugin.mtl* file specifies plug-ins that you want to load automatically. VR-Forces does not include this file in the distribution. However, *./plugins/readme.txt* explains the syntax of a *plugin.mtl* file and includes sample text. If you want to load plug-ins automatically, create a *plugin.mtl* file and put it in the *./plugins* directory.

2.2.5. The *commonObjects.mtl* File

The *commonObjects.mtl* file has mappings for objects that use the same files regardless of the symbol map file in use. The syntax of entries is documented in the file.

2.2.6. The *soundmap.mtl* File

The *soundmap.mtl* file maps sound files to munition fires and detonations. The format of entries is documented in the file.

2.2.7. The *rangeRings.mtl* Configuration File

The *rangerings.mtl* file configures how range rings are displayed in the GUI. Range rings are configured for each supported weapon. The file documents the syntax of entries.

2.2.8. The *boundingVolume.mtl* Configuration File

The *boundingVolume.mtl* file specifies the bounding volumes that VR-Forces uses in the front-end when calculating intervisibility and entity intersections are enabled. The bounding volumes specified in the object parameter database are larger than the actual dimensions of the entities because the bounding volume is used for collision avoidance calculations. You can edit the file to set the bounding volumes for intervisibility to be the actual size of entities, thereby getting more accurate intervisibility information.

The *boundingVolume.mtl* file contains bounding volumes for all entities configured in the object parameter database, either using specific entity enumerations, or through wildcard matching. If you add entities to the object parameter database, you may want to add corresponding entries to the *boundingVolume.mtl* file. The bounding volume is specified as half of the length, width, and height of the actual bounding volume. The default half-size bounding volume is two meters long, 1.5 meters wide, and 1.5 meters high.

2.3. Configuring Overlay Objects

The list of overlay objects available on the Create menu and the Overlay Palette is determined by the *overlayMenu.mtl* file. You can edit them to change the organization of objects and to add or delete overlay objects.

The overlay objects are listed on each submenu in the order that they are listed in the configuration file. Please see the comments in *overlayMenu.mtl* for an explanation of the syntax of these files.

The *overlayPalette.mtl* file configures how overlay objects are drawn. Please see the comments in the file for details.

2.4. Configuring Communication Models

Communication models are configured in the *commModelParams.mtl* file in *./data/simulationModelSets/<model-set>*. Communication models determine how radio messages travel from one entity to another. They determine which entities are connected to each other and how long it takes a message to get from one entity to another.

Communication model classes are derived from *DtCommModel* and are generated from the communication model factory based on the string types specified in the *commModelParams.mtl* file. A communication model entry contains parameters for the type of communication model descriptor and communication model class being used.

2.4.1. Simple Radio Communication Model

VR-Forces includes a default communication model called *simple-radio-comm-model*. This communication model models radio traffic in a very simple way. 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, **connection-mode**, which determines how the radio networks are set up. There are three connection-modes available:

- ♦ **aggregate** mode specifies that each aggregate has its own radio network. Messages sent from an entity can reach any entity that is in the same organization, at the same level (the entity's peers in the organization), and can also reach the entity's direct superior. All other entities are outside the network, and are unable to receive messages sent by the entity.



If an entity is not in an aggregate, it is considered to be part of the force aggregate and its radio messages go to other entities that are part of the force aggregate.

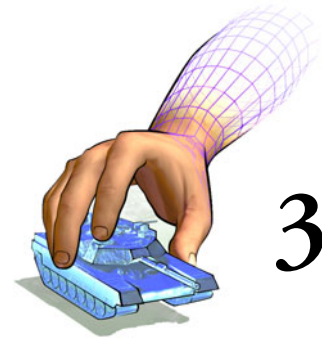
- ♦ **force** mode specifies that there is a radio network for each force in VR-Forces. Messages sent from an entity can reach any other entity of the same force.
- ♦ **all** mode specifies that all entities in VR-Forces are part of one large radio network. In this case, any entity can receive messages sent by any other entity.

2.4.2. External Communication Model

VR-Forces includes an external communication model that can communicate with an external communication effects server over the network to determine how to deliver radio messages. For details, please see Chapter 9, *Using VR-Forces with an External Communications Effects Server*.

2.4.3. 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 if radio messages get delivered. For details, please see Section 7.5.1, “[Creating Custom Communication Models](#)” in *VR-Forces Developer’s Guide*.



Scenario Files

This chapter describes the files that get created when you create a scenario.

Introduction	3-2
The Scenario File	3-3
Scenario Parameters.....	3-4
Specifying Pathnames for Scenario Files.....	3-6
Changing the Terrain Database for a Scenario	3-7
Editing the Plan Files	3-7
The Structure of an Entity Plan File	3-8
Editing the Object Map File	3-10
Editing the Order of Battle File.....	3-10
Saving Default Entity Parameters to the Order of Battle File	3-11
The Overlay File	3-11
The Selection Groups File.....	3-11
Temporary Scenario Directories	3-12

3.1. Introduction

When you create and save a scenario, VR-Forces stores information about the scenario in the following 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 sets several parameters for the scenario.
- The order of battle file, named *scenario_name.oob*, stores identification and state information for entities, control objects, and overlay objects.
- The object map file, named *scenario_name.omp*, maps objects to the back-ends on which they were created.
- The plan file, named *scenario_name.pln*, stores entity plans.
- The global plan file, named *scenario_name.gpl*, stores the scenario's global plans.
- The overlay file, named *scenario_name.ovl*, stores information about unpublished overlay objects, if any.
- The selection groups file, named *scenario_name.sgr*, stores information about selection groups, if any.

For the most part, you will not want to edit these 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.

3.2. The Scenario File

When you create a scenario, you can set its parameters in the New Scenario dialog box (Figure 3-1). Thereafter, you cannot edit scenario parameters through the GUI; however, you can edit them in a text editor.

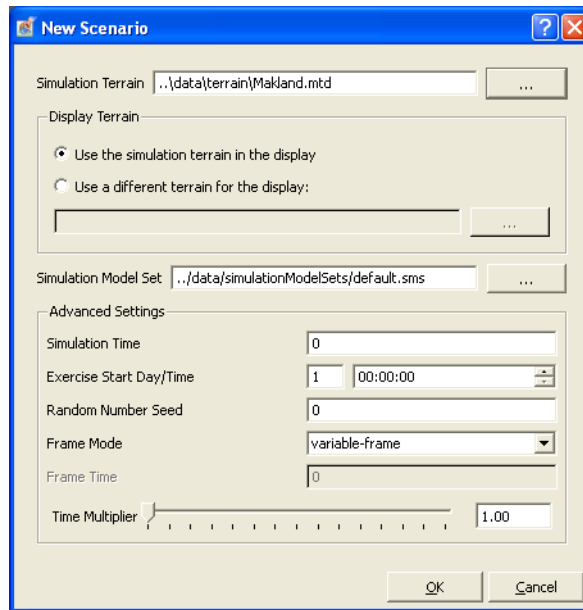


Figure 3-1. Choose Terrain dialog box

The default values for scenario settings in the New Scenario dialog box are drawn from the *default.scn* file. If you want to change the default settings, you can edit that file.

3.2.1. Scenario Parameters

A typical scenario file looks like the following:

```
(Scenario
  (Terrain-Database "..\data\terrain\Makland.mtd")
  (Gui-Terrain-Database "..\data\terrain\Makland.mtd")
  (Order-Of-Battle "..\data\scenarios\example.oob")
  (Plan "..\data\scenarios\example.pln")
  (Global-Plan "Makland.gpl")
  (Overlay "..\data\scenarios\example.ovl")
  (SelectionGroups "..\data\scenarios\example.sgr")
  (Object-Map "..\data\scenarios\example.omp")
  (time-multiplier 1.000000)
  (auto-reorganize False)
  (random-number-seed 0)
  (frame-mode "variable-frame")
  (frame-time 0.000000)
  (sim-time 0.000000)
  (exercise-start-time 0.000000)
  (additional-opds "C:/MAK/vrf3.11/data/parameters/customOpd1.opd"
    "C:/MAK/vrf3.11/data/parameters/customOpd2.opd")
  (Simulation-Model-Set-Files "..\data\simulationModelSets\default.sms")
  (version 3.000000)
)
```



- Although you can edit the **sim-time** and **exercise-start-time** parameters in scenario files, we strongly recommend that you not change these parameters for scenarios that were saved with a simulation time other than 0. Some of the state information saved in the scenario may depend on these values and if you change them, the behavior of the scenario may not be what you expect.
 - Paths in the scenario files are relative to the application directory, for example, *vrforces3.11/binDIS/vrfSim.exe*.
-

The parameters in a scenario file are as follows:

- **Terrain-Database** specifies the terrain database to be loaded by the back-end.
- **Gui-Terrain-Database** specifies the terrain database loaded in the front-end.
- **Order-Of-Battle** specifies the order of battle file.
- **Plan** specifies the plan file.
- **Global-Plan** specifies the global plan file.
- **Overlay** specifies the overlay file.
- **SelectionGroups** specifies the selection groups file.
- **Object-Map** specifies the object mapping file.
- **time-multiplier** specifies whether a scenario runs in real time or at a rate other than real time. A value greater than 1 runs the simulation faster than real time. A value between 0 and 1 runs the simulation slower 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** lets you specify the default behavior for aggregate reorganization. Set it to True or False. For information about reorganization, please see Section 3.7.1, “How Aggregates Are Organized” in *VR-Forces User’s Guide*.
- ♦ **random-number-seed** sets the default random number seed. It provides a starting point for the generation of random numbers needed by VR-Forces. Since computer-generated random numbers are not truly random, if you run a scenario repeatedly with the same random number seed (and you run in fixed-frame-mode), you will get the same results. (For more information about repeatability, please see Section 3.14.2, “Exercise Clock Modes” in *VR-Forces User’s Guide*.) Enter an integer between 0 and the largest integer permitted on your system.
- ♦ **frame-mode** sets the frame rate mode to one of the following:
 - 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 Section 3.14.2, “Exercise Clock Modes” in *VR-Forces User’s Guide*.

- ♦ **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 will prevent simulation time from advancing in either of these modes.
- ♦ **sim-time** specifies the starting simulation time, in seconds. By default, a new scenario starts with simulation time set to 0.0. A scenario saved as a checkpoint saves the simulation time of the checkpoint.
- ♦ **exercise-start-time** specifies the starting exercise time for the scenario, in seconds. By default, an exercise starts at day 1, zero hours, minutes, and seconds. However, you can set the scenario to start running with a different exercise time.
- ♦ **additional-opds** specifies custom OPD files to load (for backwards compatibility.)
- ♦ **Simulation-Model-Set-Files** specifies a list of SMS files to load.
- ♦ **version** specifies versioning information for the scenario file. Do not edit this parameter.

3.2.2. Specifying Pathnames for Scenario Files

Pathnames in a scenario file can be absolute or relative. Relative pathnames are relative to the application directory. Section 5.3.1, “[Saving Scenarios with Relative or Absolute Pathnames](#)” in *VR-Forces User’s Guide* explains how to configure the way that pathnames are saved. 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 `./binx` directory.



If you plan to run multiple front-ends and back-ends 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

The following parameters in `vrfsim.mtl` determine how pathnames for the object parameter database and terrain database are resolved:

- **defaultParameterDatabasePath** specifies the path to search for the `.opd` file 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 back-end loads a scenario file, it looks for the object parameter database using the following 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 (`vrforcesx.x/binx/vrfsim`).
3. Look in the directory specified by **defaultParameterDatabasePath**.

When the back-end loads a scenario file, it looks for the terrain database using the following 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 directory specified by **defaultTerrainDatabasePath**.
4. If defined, look in the user-defined environmental variable `VRF_TERRAIN_PATH`.

3.2.3. Changing the Terrain Database for a Scenario

When you create a new scenario, you must choose terrain databases for the front-end and back-end. By default, the back-end uses the same database as the front end. Please see Section 3.12.1, “[Using Large Terrain Databases](#)”, in *VR-Forces User’s Guide* for details.

Once you have chosen the terrain database, you cannot change it from the graphical user interface. You can edit the pathname of a terrain database in the scenario file. However, unless the new terrain database covers exactly the same geographic area as the old one, the order of battle and overlay information associated with the scenario may be confusing or useless.

Although you would not typically change the terrain database used for a scenario, if you create a scenario in the 2D GUI and want to use it in the 3D Front End (and vice versa), you may need to change the terrain specified by the **Gui-Terrain-Database** parameter so that the front-end uses the appropriate type of database. For more information, please see the sections that describe creating scenarios in *VR-Forces User’s Guide* and *VR-Forces 3D Front End User’s Guide*.

3.3. Editing the Plan Files

VR-Forces saves two plan files for each scenario, an entity plan file (*.pln*), which stores the plans for all individual entities, and the global plans file (*.gpl*), which stores global plans.

Plan files are created interactively in the VR-Forces graphical user interface. Using the GUI to create and edit plans ensures that the plan is written using the correct syntax. We recommend that you not try to create plan files or edit plan files by hand. However, there may be occasions when it makes sense to edit a plan file by hand or programmatically. For example, if you change the name of an entity, using the GUI to change all plan statements that refer to that entity could be tedious. It would be much quicker to open the plan file in a text editor and replace appropriate instances of the old entity name with the new entity name.



Before you edit a plan file, make a backup copy that you can revert to if necessary.

3.3.1. The Structure of an Entity Plan File

Figure 3-2 illustrates a simple entity plan that specifies one task for each of two entities (a global plan has a similar syntax):

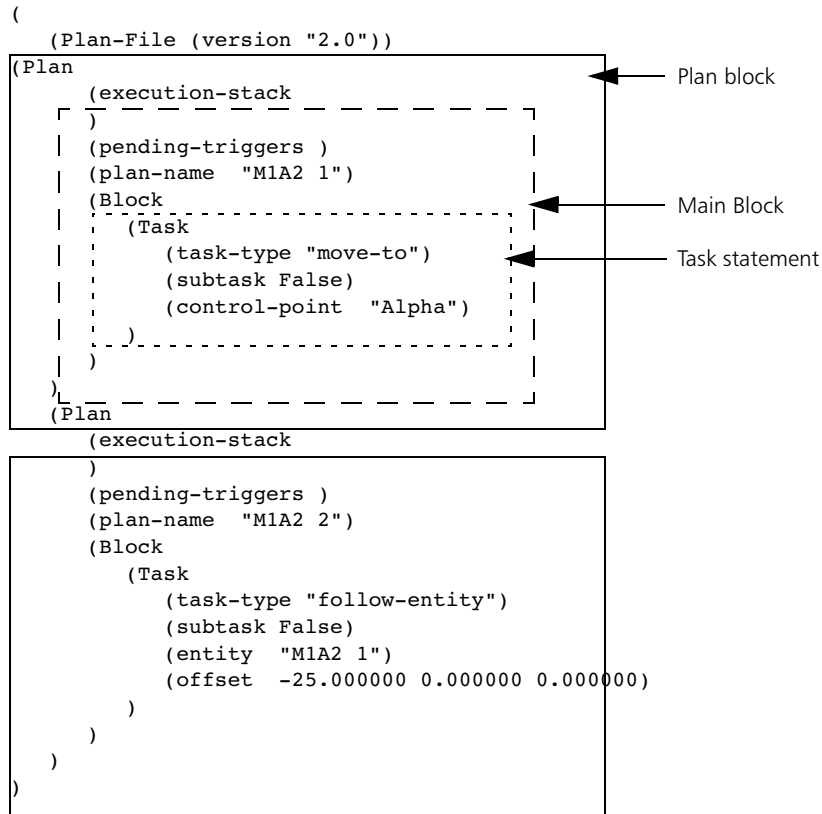


Figure 3-2. Simple plan

Plans are specified for entities by entity name. The plan is specified within a Plan block. The content of the plan is specified in the Main block. Tasks and set statements are specified in the Main block and in conditional statements. Conditional statements can be specified in the main block and can be nested in other conditional statements.

The basic syntax for a set statement in a plan is:

```

(Block
  (Set
    (set-data-request-type "set-heading")
    (heading 0.872665)
  )
)
  
```

Figure 3-3 illustrates the basic syntax for a conditional expression. It includes a task statement in the main block to show relative indentation.

```

(Block ← Main block
  (If ← If statement
    (Condition
      (EntInArea
        (Entity "M1A2 1")
        (Area "Blue")
        (Modifier ",,"))
      )
    )
    (Then-Block ← Then block
      (Task
        (task-type "move-to")
        (subtask False)
        (control-point "Bravo")
      )
    )
    (Else-Block ← Else block
      (Task
        (task-type "move-to")
        (subtask False)
        (control-point "Alpha")
      )
    )
  )
  (Task ← Task statement
    (task-type "follow-entity")
    (subtask False)
    (entity "M1A2 1")
    (offset -25.000000 0.000000 0.000000)
  )
)

```

Figure 3-3. Conditional expression with subblocks

To edit a plan file by hand, add Task, Set, If, While, and When sections as needed to the Block section for each echelon ID. For tasks or set statements, you will always have a **task-type** or **set-data-request-type** parameter that needs to contain the type of task or set data request. The presence of additional parameters depends on the task or set data request that you created. You will need to know the names of the parameters and what type of value to specify.

Make sure that you match parentheses properly.

3.4. Editing the Object Map File

Mappings between objects and back-ends are saved to the object map file. If you have a scenario that uses multiple back-ends 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. The following example shows two entities and a control object mapped to two different back-ends:

```
(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 back-end on which the object was simulated.

If you want to run an existing scenario using back-ends 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 back ends. For details about load balancing, please see Section 5.2.2, “[Load Balancing a Scenario](#)” in *VR-Forces User’s Guide*.

3.5. Editing the Order of Battle File

VR-Forces saves entity state information and graphical object information to the order of battle file.

If you save a scenario after the simulation has run, for example, as a checkpoint, the order of battle file stores the state information for the entities at the time of the save. The next time you load this scenario, VR-Forces creates the entities using the settings in the object parameter database, but then overrides those settings with the information in the order of battle file. Therefore, changes you make to the object parameter database might not get picked up by existing scenarios. If you want to change a basic entity value, you might need to edit the order of battle file.

3.5.1. Saving Default Entity Parameters to the Order of Battle File

When an entity in a scenario is saved to the order of battle (OOB) 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 entity's object parameter database entry. The practical effect of the USE-DEFAULT keyword in OOB files is that if you modify the object parameter database, the next time you load a scenario those changes can affect the entities and objects in the scenario.

This lets you experiment with changes to entity parameters and view the effects of the changes to existing scenarios, without modifying the order of battle files.



- You can edit the OOB file manually to override the value obtained from the object parameter database.
 - If successive scenarios use the same object parameter database, VR-Forces does not, by default, reload the object parameter database before running each scenario. If you want VR-Forces to reload the object parameter database, (for example, if you are making changes to it and testing the changed behaviors) set the `forceParameter-DbReload` parameter to 1.
-

3.6. The Overlay File

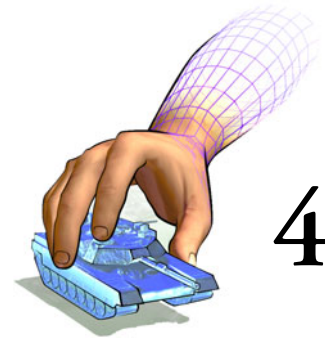
The Overlay file (*.ovl*) saves information about unpublished overlay objects and the order in which overlays and overlay objects should be drawn in the GUI.

3.7. The Selection Groups File

The Selection Groups file (*.sgr*) saves information about selection groups. The information saved does not lend itself to editing by hand.

3.8. Temporary Scenario Directories

When you load a scenario, VR-Forces creates temporary directories in which it stores files that it needs 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 front-end, the temporary directory is created by *DtVrfRemoteController* and starts with the prefix *vrf_{fe}*. In the back-end, the temporary directory is created by *DtCgfDispatcher* and has the prefix *vrf_{be}*. In both cases, the temporary directory is removed when VR-Forces shuts down.



Merging Scenarios

This chapter explains how to merge scenarios with the Scenario Merge tool.

The Scenario Merge Tool	4-2
Starting Scenario Merge	4-2
Creating a Scenario Merge Project.....	4-3
Adding Scenarios to a Project	4-3
Removing Scenarios from a Project.....	4-3
Specifying an Ignore List File.....	4-4
Saving a Project	4-5
Loading a Project.....	4-5
Merging Scenarios.....	4-6
Displaying Scenario Files	4-6
Editing Scenario Files	4-7
Closing Scenario Files.....	4-7
Specifying the Output Notification Level.....	4-7
Using the Scenario Merge Command-Line Interface.....	4-8

4.1. The Scenario Merge Tool

VR-Forces users often assign scenario development to multiple persons or teams. Then they merge the separately-developed scenarios into a master scenario. Merging scenarios by hand can be a difficult task due to the requirement to reconcile entity and object names, plans, and other scenario elements that could be duplicated, particularly if objects are named using the default conventions.

The Scenario Merge tool allows you to combine two or more VR-Forces scenarios into a single scenario. Scenario Merge detects and resolves conflicts so that the simulated objects can co-exist in a single scenario, without losing their unique identity. Likewise, references to conflicting names are resolved, so that the plans from both scenarios will work properly.

Scenario Merge has a graphical user interface and a command-line interface. It is an off-line tool for combining existing scenarios. You cannot merge scenarios from within the VR-Forces GUI.

4.2. Starting Scenario Merge

The graphical user interface is the primary way to use the Scenario Merge tool.

- To start Scenario Merge in Windows, on the Start menu, choose Programs → MAK Technologies → VR-Forces *x.x* → Tools → Scenario Merge.

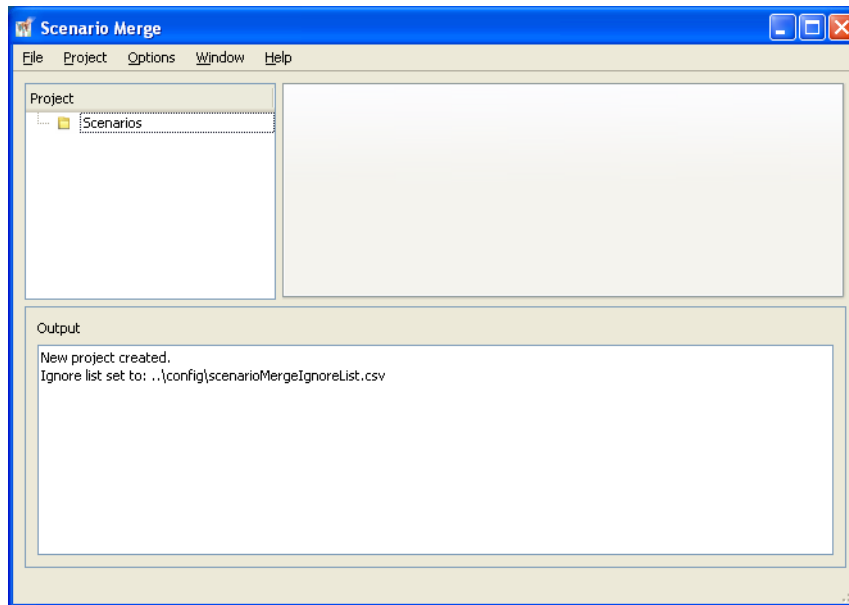


Figure 4-1. Scenario Merge window at startup

- To start the Scenario Merge GUI from the command line, enter the following:
`scenarioMerge [ProjectFile]`

If you start Scenario Merge from the command line and specify *projectFile*, Scenario Merge launches the GUI, then loads the project file. Otherwise, it opens a new project. Command-line syntax is described in [“Using the Scenario Merge Command-Line Interface,”](#) on page 4-8.

4.3. Creating a Scenario Merge Project

Before you can merge scenarios, you must create a Scenario Merge project. A project identifies the scenarios to merge, where to write the merged scenario, and optionally, a list of attributes to ignore during the merge process.

When you start Scenario Merge, it automatically creates a new, empty project. However, you can also start a new project manually.

- To open a new project, choose File → New Project.

4.3.1. Adding Scenarios to a Project

You can add as many scenarios as you want to a project. However, the first scenario you add is the key scenario. It determines the parameters of the final scenario, including, terrain database, object parameter database, exercise start time, and so on. If any of the scenarios that you merge into the first scenario have different scenario parameters, they are ignored and a diagnostic message is recorded in the project log file.

If you add scenarios from different terrains, Scenario Merge uses only geocentric coordinates for entity position data. This is accomplished using the Ignore List File, which is explained in [“Specifying an Ignore List File,”](#) on page 4-4.



Before you add a scenario created in VR-Forces 3.10.x or earlier to a project, open it in the current version of VR-Forces and save it.

To add a scenario to a project, use one of the following methods:

- ♦ Choose Project → Add Scenario, and select a scenario to add.
- ♦ Drag a scenario file (*.scn*) from Windows Explorer to the Scenario Merge Project tree view.
- ♦ Drag a scenario folder from Windows Explorer to the Scenario Merge Project tree view. If you drag a folder, Scenario Merge adds all scenario files in the folder to the project.

4.3.2. Removing Scenarios from a Project

To remove a scenario from a project:

1. In the Project window, select the scenario that you want to remove.
2. Choose Project → Remove Scenario.

4.3.3. Specifying an Ignore List File

Sometimes it is useful to suppress values when writing to the final scenario. For example, Scenario Merge suppresses any reference to a local coordinate system in the Order of Battle file (*.oob*) file. You can specify values to ignore in an ignore list file. The ignore list file is a comma-delimited text file, for example:

```
is local-kinematics-state,IGNORE
local-vertices,IGNORE
```

The first value is the string to search for during a merge operation. The second value is a text string that will replace the key in the merged scenario.

If a substitution is specified in the ignore list, it is applied to the following files in the scenario:

- ♦ Order of Battle (*.oob*)
- ♦ Plan (*.pln*)
- ♦ Global Plan (*.gpl*)
- ♦ Object Map (*.omp*).

To create an ignore list file, you must use a text editor. The Scenario Merge GUI just lets you specify which file to use. You must include the following lines from the default ignore list:

```
local-kinematics-state,IGNORE
local-vertices,IGNORE
```

These entries are needed when merging scenarios that use different terrains.

To specify the Ignore List file:

1. Choose Project → Specify Ignore List. The Specify Ignore File dialog box opens.
2. Type a filename or click the Browse button and select the Ignore List file.
3. Click OK.

4.3.4. Saving a Project

When you save a project, Scenario Merge creates a project file. You can load the project at a later date to merge the scenario again or change which scenarios you want to merge. You can also use a project file to merge files from the command-line. For details about running from the command-line, please see [“Using the Scenario Merge Command-Line Interface,”](#) on page 4-8.

To save a project:

1. Choose File → Save Project. The Scenario Merge dialog box opens.
2. Select a directory and type a name for the project.
3. Click Save. The project gets saved with the extension *.smp*.

The Scenario Merge Project File

A merge project is a comma-delimited text file that lists the scenarios to merge, the desired location for the final result, and merge options. The following is an example of a Scenario Merge project file:

```
OUT,..\\data\\scenarios\\demoMerge
IGN,..\\config\\scenarioMergeIgnoreList.csv
SCN,..\\data\\scenarios\\scenarioOne\\scenarioOne.scn
SCN,..\\data\\scenarios\\scenarioTwo\\scenarioTwo.scn
SCN,..\\data\\scenarios\\scenarioThree\\scenarioThree.scn
SCN,..\\data\\scenarios\\scenarioFlour\\scenarioFlour.scn
```

Each line starts with one of the following keywords:

- ♦ OUT specifies the location for the output scenario files.
- ♦ IGN specifies an Ignore List file. For details, please see [“Specifying an Ignore List File,”](#) on page 4-4.
- ♦ SCN specifies the path of a scenario to merge in.



Paths are relative to the location of *scenarioMerge.exe*.

4.3.5. Loading a Project

If you have saved a project, you can load it, edit it as necessary, and regenerate the merged scenario.

To load a project:

1. Choose File → Load Project. An Open dialog box opens.
2. Select the Scenario Merge project file that you want to load.
3. Click Open.

4.4. Merging Scenarios

- To merge the scenarios in a project, choose Project → Merge. The scenarios are merged and informational messages are displayed in the console.

4.5. Displaying Scenario Files

You can display the text of one or more of the scenario files in a project.

- To display a scenario file, in the Project window, double click the file. It is displayed to the right of the Project window.

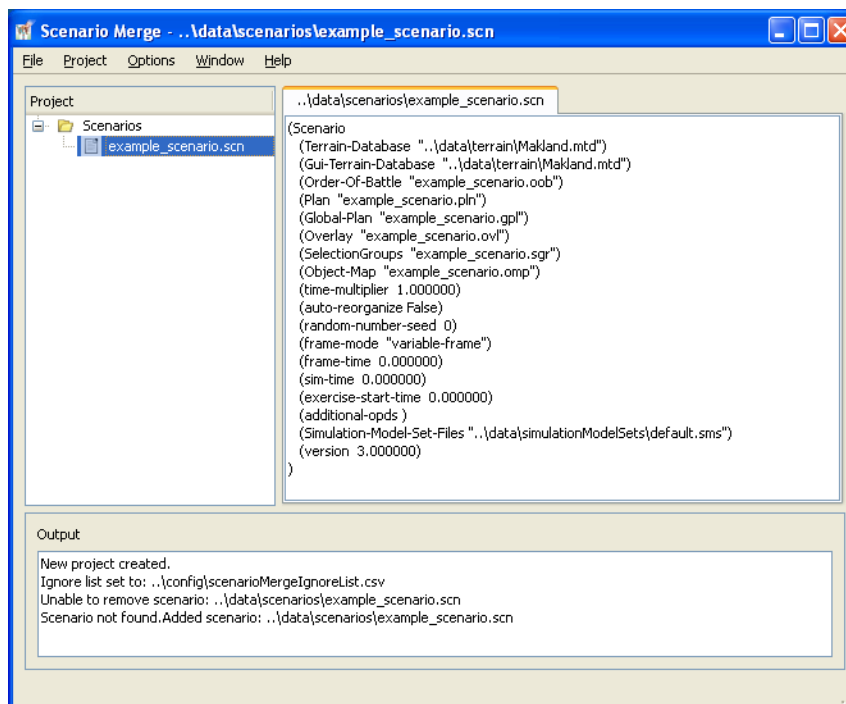


Figure 4-2. Scenario file display

4.5.1. Editing Scenario Files

If you display a scenario file, you can edit it. Any edits you make are saved back to the original scenario. You cannot edit the scenario file that gets created when you merge scenarios.



Editing scenario files is risky to scenarios and is advisable only for experienced VR-Forces users. Be sure you fully understand the contents of the file as described elsewhere in this chapter and VR-Forces documentation.

To edit a scenario file:

1. In the scenario file display window, make any edits desired.
2. Choose File → Save Scenario or Save All.

4.5.2. Closing Scenario Files

To close a scenario file that is displayed by Scenario Merge, use one of the following methods:

- ♦ Choose Window → Close. If multiple files are open, this closes the most recently opened one.
- ♦ Choose Window → Close All.
- ♦ Choose File → Close Scenario.

4.6. Specifying the Output Notification Level

You can specify the notification level used for sending messages to the output console. The notification levels correspond to the notification levels used throughout MÄK products and described in Chapter 2, [VR-Forces Configuration Files](#).

- To set the notification level, choose Options → Set Notify Level → *level*, where *level* is one of the following:
 - ♦ Fatal
 - ♦ Warning
 - ♦ Info
 - ♦ Verbose
 - ♦ Debug.

4.7. Using the Scenario Merge Command-Line Interface

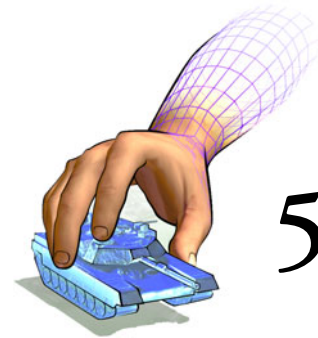
To run the Scenario Merge tool from the command line, use the following syntax:

```
scenarioMerge -c projectFile
```

where:

-c specifies that Scenario Merge should run in command-line mode.

projectFile specifies a Scenario Merge project file to load. For details about creating project files, please see [“Adding Scenarios to a Project,”](#) on page 4-3.



Editing Entity Models

The Entity Editor is a stand-alone GUI tool that lets you configure entities, edit the Create menu, and create new simulation model sets.

Introduction	5-3
Simulation Model Sets.....	5-4
Starting the Entity Editor	5-5
Creating a New Simulation Model Set	5-6
Specifying the Default Simulation Model Set for Scenarios	5-6
Opening a Simulation Model Set	5-7
Editing an Entity Model	5-8
Changing an Entity's Icon	5-8
Editing an Entity's Category	5-11
Editing an Entity's Force.....	5-12
Changing an Entity's Entity Enumeration	5-13
Changing an Entity's Name	5-13
Specifying Whether an Entity can be Created.....	5-14
Editing Behavioral Parameters	5-14
Undoing (Reverting) Changes to an Entity	5-15
Editing an Entity's Systems	5-15
Adding A System	5-16
Selecting a Damage Model or Movement Model	5-17
Editing a System's Properties.....	5-17
Removing a System	5-17
Editing the Create Menu.....	5-19
Generating the Create Menu Automatically.....	5-19
Editing the Create Menu Manually	5-20
Undoing (Reverting) Changes to the Create Menu.....	5-22

Editing Entity Models

Creating a New Entity	5-18
Deleting an Entity Model from a Model Set.....	5-19
Editing the Category List.....	5-22
Adding a New Category	5-22
Removing a Category	5-23
Renaming a Category	5-23

5.1. Introduction

Many VR-Forces users want to edit some or all of the following entity characteristics:

- ♦ Weapons systems
- ♦ Display image
- ♦ Basic parameters
- ♦ Sensors
- ♦ Force.

Users also want to add new entities and rearrange how entities are organized on the Create menu. The VR-Forces Entity Editor allows you to edit most of the aspects of an entity that a typical VR-Forces user will want to change. For example, you can change the force and icon associated with an entity type. You can easily add and remove sensors, weapons systems, and other major component systems, and can edit their properties.

The Entity Editor does not let you add or remove individual components or edit most of the more detailed entity parameters. For these advanced editing tasks, you can use the OPD Editor, which is described in Chapter 6, [Advanced Entity Model Editing](#), or edit specific configuration files, which are described throughout this manual.

If you want to add new entities or customize the entities supplied with VR-Forces, start with the Entity Editor. Use the OPD Editor or a text editor only if the Entity Editor does not let you make the changes you need to make.



Do not try to edit an entity model in the Entity Editor and the OPD Editor at the same time. Use one tool, save the changes, and close the tool before opening the other.

5.1.1. Simulation Model Sets

A simulation model set (SMS) defines all of the entity and 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 entities and other simulation objects. These files include the object parameter database, damage files, formation files, and symbol mapping files. Every scenario must reference one or more simulation model sets.

If you use the Entity Editor to edit entity 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 Entity Editor.



When you use the Entity Editor, you are editing a simulation model set. The default SMS provided with VR-Forces is *default.sms*. MÄK recommends that you keep the default SMS as a reference model set and not edit it. If you want to edit entity models, create a new SMS based on *default.sms* and use it as the default SMS for your scenarios. For details about creating a new SMS and changing the default SMS, please see [“Creating a New Simulation Model Set,”](#) on page 5-6.

5.1.2. Starting the Entity Editor

- To start the Entity Editor in Windows, choose Start → Programs → MÄK Technologies → VR-Forces *x.x* → Tools → Entity Editor. The Entity Editor opens (Figure 5-1).

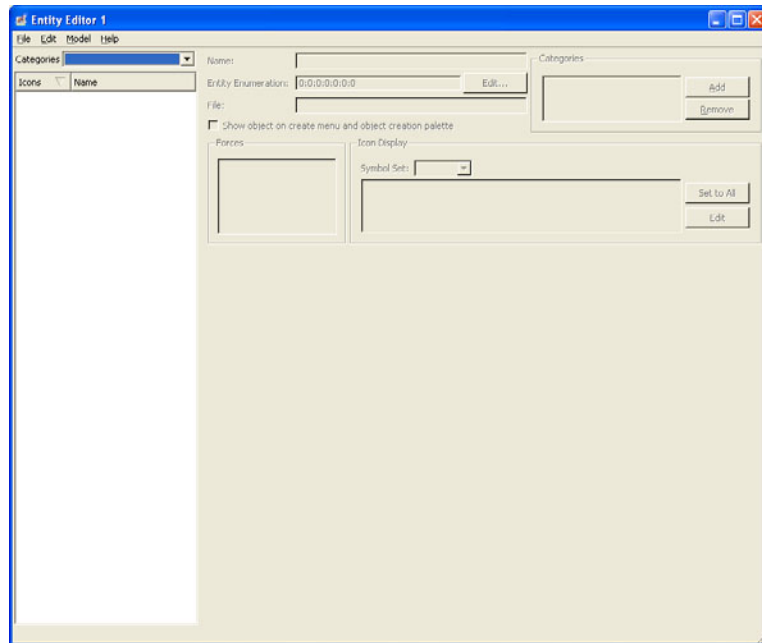


Figure 5-1. Entity Editor

To start the Entity Editor from the command-line in Windows or Linux:

1. Change directory to `./bin`.
2. Run:

```
entityEditor
```

5.2. Creating a New Simulation Model Set

You can create new SMSs by copying and editing an existing model set or you can create a completely new model set and create the entities that you want to include in it.

To create a new simulation model set from an existing model set:

1. Choose File → New Simulation Model Set. The New Simulation Model Set dialog box opens.
2. Select Create From Existing Simulation Model Set.
3. Click Next. The Select Model Set to Copy dialog box opens.
4. Select a model set and click Next. The New Model Set Name dialog box opens.
5. In the File name box, type a name for the new model set.
6. Click Finish. The Entity Editor loads information from the source model set.
7. Make any changes you want to the model set.
8. Choose File → Save Simulation Model Set.

To create a completely new simulation model set:

1. Choose File → New Simulation Model Set. The New Simulation Model Set dialog box opens.
2. Select Create New Simulation Model Set.
3. Click Next. The New Model Set Name dialog box opens.
4. In the File name box, type a name for the new model set.
5. Click Finish. The Entity Editor loads systems information from the default model set.
6. Create the entities that you want to have in the model set, as described in [“Creating a New Entity,”](#) on page 5-18.
7. Choose File → Save Simulation Model Set.

5.2.1. Specifying the Default Simulation Model Set for Scenarios

When you create a new scenario, the New Scenario dialog box lists *default.sms* as the SMS to use. You can change the default SMS by editing the `simulationModelSet` parameter in *vrfsim.mtl*.

5.3. Opening a Simulation Model Set

To edit entity models in the Entity 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. The SMS is loaded and a list of entities is displayed in the Entity Editor.

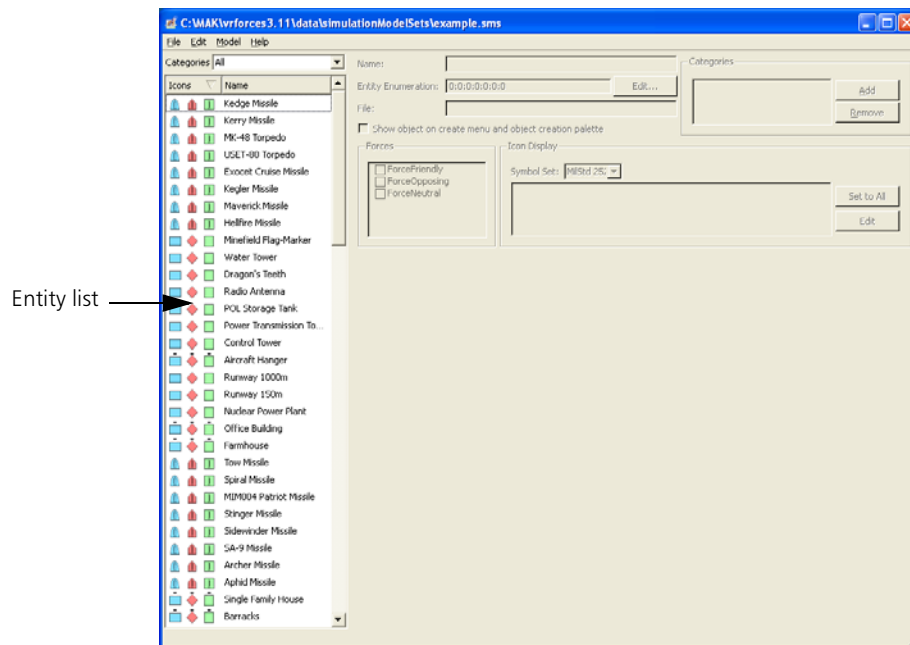


Figure 5-2. Entity Editor with model set loaded

5.4. Editing an Entity Model

This section explains how to edit an existing entity model. You can:

- ♦ Change an entity's icon
- ♦ Change the force assignments
- ♦ Add and remove systems
- ♦ Edit the properties of systems
- ♦ Edit some entity parameters
- ♦ Change the entity's category
- ♦ Change the entity's name
- ♦ Specify whether or not the entity can be created from the GUI.



When you edit an entity model in the Entity Editor, you are essentially editing the OPE file that it uses. In the default SMS, some OPE files are used by more than one entity. If you edit a particular entity in the Entity Editor, the tool creates a unique OPE file for that entity, so the changes do not affect other entities that might be using the original OPE file.

5.4.1. Changing an Entity's Icon

Each object can be created as a member of one or more forces. For each force that it can be created under, it has an image that is displayed in the map and various menus and lists. You can change the image associated with each object for each force it is associated with. You can change the image by selecting the symbol set to use or you can select a specific image. You can mix the types of images used from entity to entity and force-to-force for a particular entity.

To change the symbol set for an entity's image:

1. Select the entity whose icon you want to change. Its attributes are displayed (Figure 5-3).

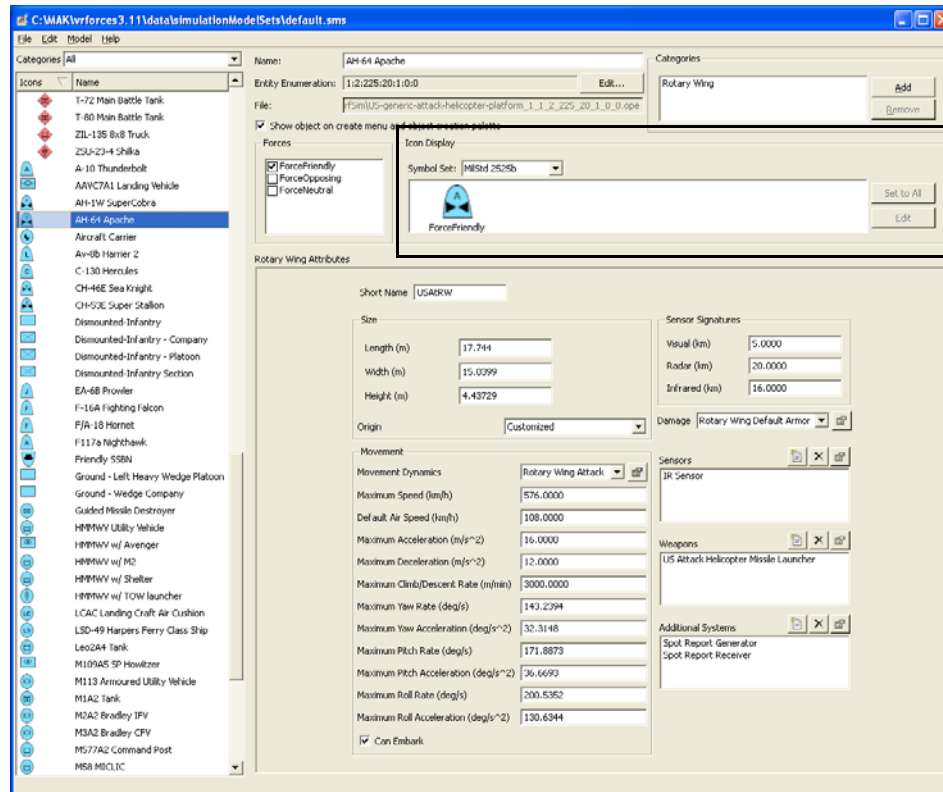


Figure 5-3. Attribute display for an AH-64 Apache

2. In the Icon Display group box, select an option from the Symbol Set drop-down list.

i

If you specify a specific image for an object, changing the symbol set does not affect the assigned image.

To specify an image for an entity:

1. Select the entity in the entity list.
2. In the Icon Display group box, click the image that you want to change.
3. Click the Edit. The Edit Icon dialog box opens.

4. Select an option from the Icon Type drop-down list. The Edit Icon dialog box redisplay to show the options appropriate to that icon type, as follows:
 - Animated Image: The Edit Icon dialog box displays a list of the images used to create the animated image. You can add to this list, remove files, change their order, or modify (replace) them. You can change how long each image is displayed (frame delay) as the animation is created.
 - Unicode True Type Fonts (2525B): You can change the echelon indicator, fill, and outline used for the image (Figure 5-4).

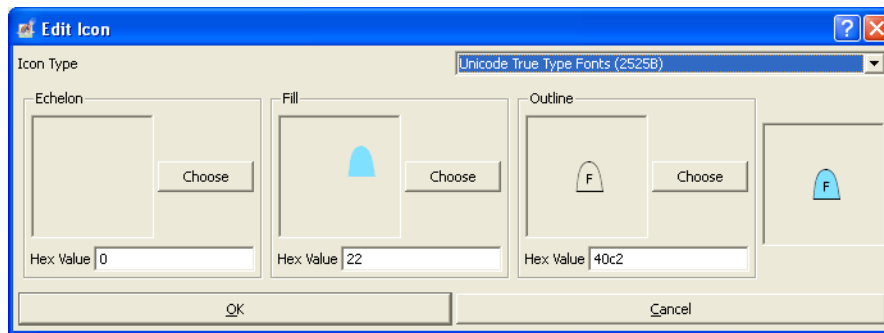


Figure 5-4. Edit Icon dialog box

- Image: You can select a raster image to use as the object icon.
 - Unicode (Extended) True Type Fonts (2525B): You can change the outline and the comment associated with the image.
5. Click OK. The new image is displayed in the Icon Display window.

Using the Same Image for an Entity Regardless of Force

Ordinarily, the icons for entities in different forces are distinguished by color. For images, there may be differences in color or design. However, if you want to, you can use the same image for an entity regardless of its force.



This procedure is not valid for MilStd 2525b images.

To use the same image for an entity type regardless of its force:

1. In the Icon Display group box, select the image you want to use for all forces.
2. Click Set to All.



To use this procedure, you must select two or more forces for the entity.

Adding New Symbol Sets

VR-Forces includes four symbol sets. You can add additional symbol sets and have them appear in the Entity Editor.

To add a symbol set to the Entity Editor:

1. Open the SMS file in a text editor. (SMS files are in *./data/simulationModelSets*.) The file contains a list of symbol map files similar to the following:

```
(symbol-map-files
  (symbol-file "MilStd 2525b" "$(model-set-directory)\gui\symbolmap-
    ttf.mtl")
  (symbol-file "MilStd 2525a" "$(model-set-directory)\gui\symbolmap-
    bmp.mtl")
  (symbol-file "Images" "$(model-set-directory)\gui\symbolmap-
    images.mtl")
  (symbol-file "Perspective Images" "$(model-set-
    directory)\gui\symbolmap-images-perspective.mtl")
)
```

2. Add a symbol-file entry for each symbol map file that you want to use. For details about symbol map files, please see [“Symbol Mapping Files,”](#) on page 2-19.

5.4.2. Editing an Entity’s Category

Entities are organized by categories on the Create menu and the Entity Palette. You can change the category that an entity is listed in. You can also list it in more than one category.

To change an entity’s category:

1. In the Categories group box, click the category that you want to change. The field becomes editable.
2. Select a different category from the drop-down list.

To add a creation category:

1. In the Categories group box, click Add. A new entry is added to the Categories list.

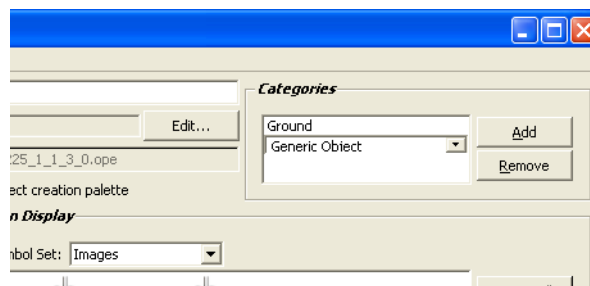


Figure 5-5. Adding a category

2. Select a category from the drop-down list.

5.4.3. Editing an Entity's Force

Each entity can be created as a member of one or more forces. You can specify which force an object can be created for. When you add an entity to a force, an icon for that force is assigned. If you remove an entity from a force, the icon is removed. You can quickly see which forces an entity is associated with by scanning the entity list in the Entity Editor (Figure 5-6).

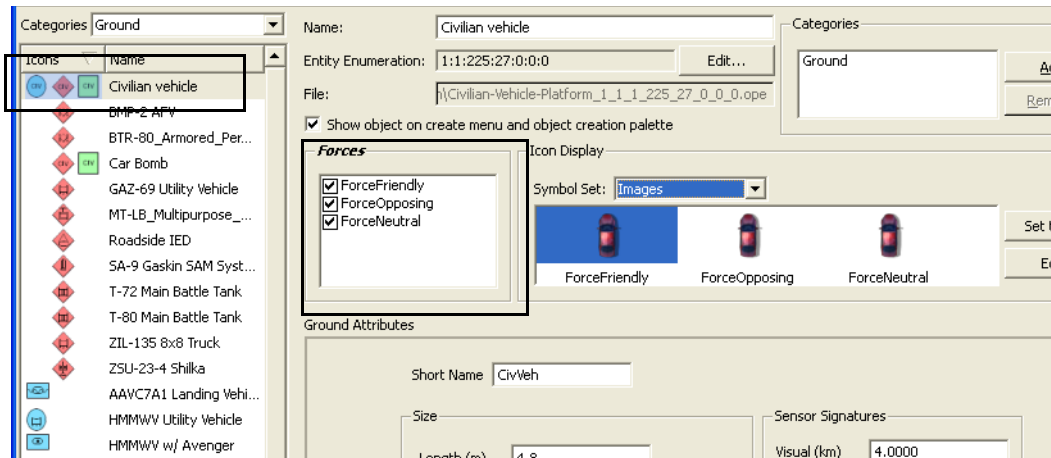


Figure 5-6. Entity (civilian vehicle) assigned to multiple forces

- To make or remove force assignments for an entity, in the Forces list, select or clear the check boxes for the forces that you want the entity to be assigned to.

5.4.4. Changing an Entity's Entity Enumeration

The Entity Editor lets you change an entity's enumeration. You might want to or have to do this for any of the following reasons:

- ♦ If you create a new model, you must specify an entity enumeration.
- ♦ If you create a new model from an existing model, the Entity 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 Entity Editor generates.
- ♦ You might want to change an existing entity's enumeration.



When you change an entity's enumeration and save the change, the Entity Editor creates a new OPE file with the new enumeration in the file name. It does not delete or change the OPE file with the entity's previous enumeration.

To change an entity's entity enumeration:

1. Select the entity in the entity list.
2. Click Edit next to the Entity Enumeration text box. The Change Enumeration dialog box opens (Figure 5-7).

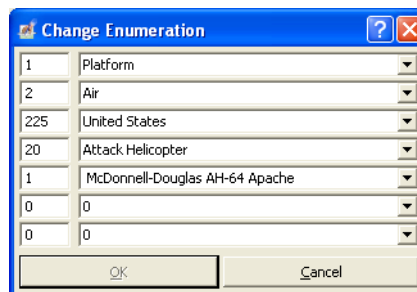


Figure 5-7. Change Enumeration dialog box

3. Edit the enumeration by typing new enumerations in the left column or selecting new categories in the second column.
4. Click OK. The enumeration changes and the OPE filename changes.

5.4.5. Changing an Entity's Name

To change an entity's name:

1. Select the entity in the entity list.
2. Type a new name in the Name text box.

5.4.6. Specifying Whether an Entity can be Created

You can control whether or not an entity is listed in the Create menu and Entity Palette. It may not make sense to create some types of entities, such as missiles, which are normally added to a scenario and removed dynamically.

- To enable or disable creation of an entity from the Create menu and Entity Palette, select or clear the Show Object on Create Menu and Object Creation Palette check box (Figure 5-8).

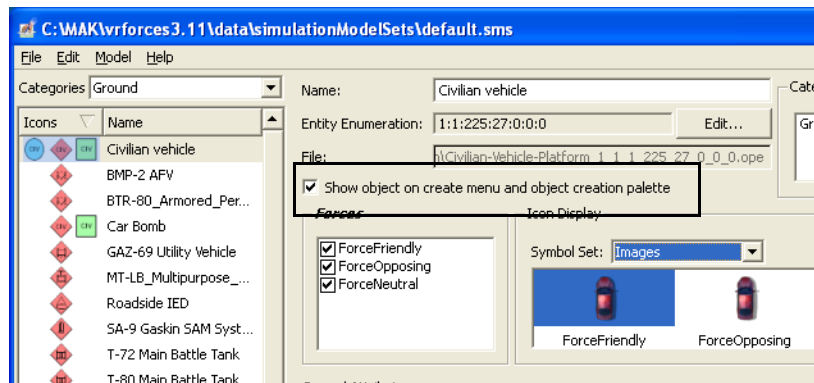


Figure 5-8. Show Object on Create Menu and Object Creation Palette check box

5.4.7. Editing Behavioral Parameters

The Entity Editor lets you edit some of the entity parameters that affect behavior.



The Sensor Signature parameters specify the distance at which other entities can sense the entity you are editing. By contrast, the sensor systems on the entity control its ability to sense other entities. For more information, please see [“Editing an Entity’s Systems,”](#) on page 5-15.

- To edit entity parameters, edit any of the parameters in the Attributes panel. (The panel reconfigures itself as you select different types of entities.)

5.4.8. Undoing (Reverting) Changes to an Entity

If you have made changes to an entity and have not yet saved the simulation model set, you can revert to the original entity configuration. You can revert an entity at any time prior to saving the SMS, regardless of how many other entities you may have changed since you edited the entity you want to revert.

To undo changes to an entity:

1. Select the entity in the entity list.
2. Choose Model → Revert Model.

5.5. Editing an Entity's Systems

Entities have complex capabilities, such as the ability to move, to take damage, to sense other entities, and to fire munitions. These capabilities are organized as systems. You can quickly change how an entity functions by adding, removing, and changing its systems. The Entity Editor lists the systems available in each category and only lists systems that are appropriate for the entity type (Figure 5-9). For example, a lifeform can have an M-16 or an RPG, but not a cruise missile.

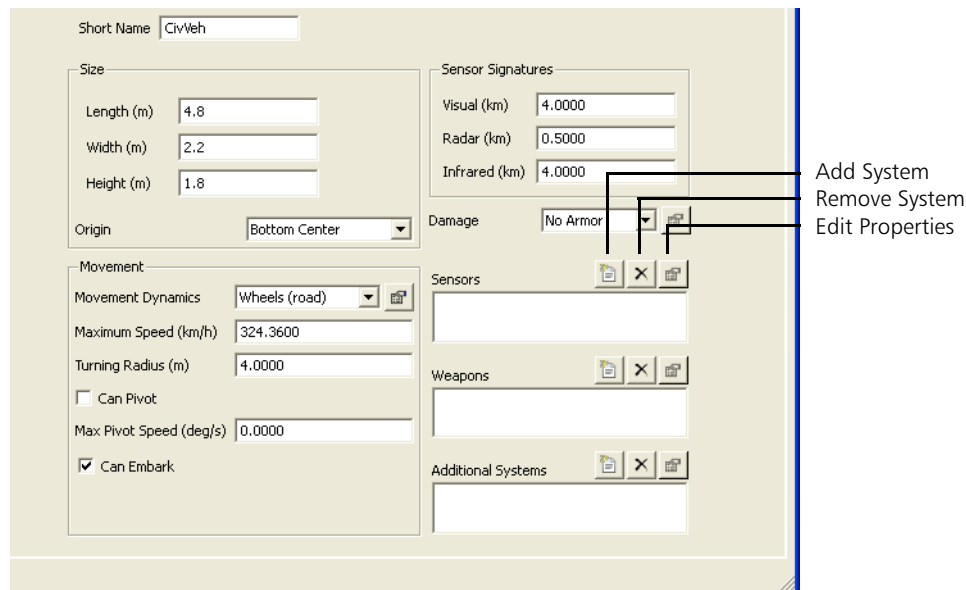


Figure 5-9. Entity systems

Some systems are interdependent. For example, if you add a weapon to an entity, it will not be able to use it unless it also has a sensor system so that it can detect hostile entities. 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.

The Entity Editor lets you edit the following types of systems:

- ♦ **Damage.** You can specify the type of damage model that an entity uses. When you select a damage model, the Entity Editor configures the damage files that are appropriate for the entity type.
- ♦ **Sensors.** You can specify the sensors that an entity uses to locate other entities in the simulation. (By contrast, sensor signatures, which you can configure in the parameter section of the window, specify the distance at which other entity sensors can locate this entity.) When you add a weapon, the Entity Editor configures the components and connections necessary for the sensor to work.
- ♦ **Weapons.** The Entity Editor lists the weapons systems that are appropriate for the vehicle type. When you add a weapon, the Entity Editor configures the components, targets, ammunition, and connections necessary for the weapon to work.
- ♦ **Movement.** You can specify the type of movement system that an entity uses, for example, tracked, wheeled, pedestrian, and so on. When you select a movement system, the Entity Editor configures all of the actuators and controllers that the entity needs to implement the system.
- ♦ **Other systems.** This category includes spot reports and IFF transponders.

5.5.1. Adding A System

There is no specific limit to the number of sensors, weapons, or other systems that you can configure for an entity.

To add a system:

1. Click the Add System button for the system type you want to add (Figure 5-9). The Add System dialog box opens (Figure 5-10). It lists the systems of this type that are available for the entity you are editing. If you select a system in the window, the dialog box displays a brief description.

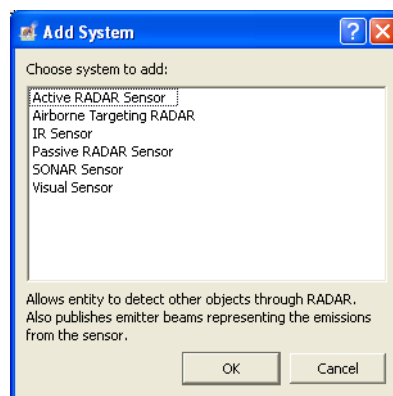


Figure 5-10. Add System dialog box

2. Select the system you want to add.

3. Click OK. The system is added to the list in the Entity Editor.

5.5.2. Selecting a Damage Model or Movement Model

You can only have one damage model and movement model specified for an entity.

To specify a damage or movement model:

1. Select an option from the Damage drop-down list.
2. In the Movement group box, select an option from the Movement Dynamics drop-down list.

5.5.3. Editing a System's Properties

Some systems have properties that you can edit.

To edit a system's properties:

1. 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.
2. Click the Properties button (Figure 5-9). A Properties dialog box opens.
3. Edit the properties as desired.
4. Click OK.

5.5.4. Removing a System

You can remove sensor, weapons, 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. Select the system you want to remove.
2. Click the Remove System button (Figure 5-9).

5.6. Creating a New Entity

You can create a new entity by using an existing entity as a template or by building a completely new entity.

To create a completely new entity:

1. Choose Model → New Model. The Create New Entity dialog box opens.
2. Select an entity category from the Entity to Create drop-down list.
3. Type a name for the entity.
4. Click OK. The Entity Editor adds a new entry to the entity list. The editor shows default values for all the fields.
5. Specify an enumeration for the entity.
6. Specify symbol set and force values for the new entity.
7. Specify appropriate values for the entity parameters.
8. Type a name to be used in the entity's label in the Short Name text box.
9. Specify a movement system.
10. Specify a damage model.
11. Optionally, add weapons systems.
12. Optionally, add sensors.
13. Save the simulation model set.

5.6.1. Creating a New Entity from an Existing Entity

If you create a new entity from an existing entity, the new entity starts with all of the behavioral characteristics and systems of the original entity. It increments the entity enumeration. You just have to name it and customize it for the new entity.

To create a new entity from an existing entity:

1. In the entity list, select the source entity.
2. Choose Model → New Model from Existing. The Create New Entity dialog box opens.
3. Type a name for the entity.
4. Click OK. The new entity is added to the entity list. The enumeration is incremented and the an OPE file name is created based on the entity name.
5. Edit the parameters to provide the new entity the characteristics you want for it.
6. Save the simulation model set.

5.7. Deleting an Entity Model from a Model Set

To delete an entity model from a simulation model set:

1. Select the entity in the entity list.
2. Choose Model → Delete Model or press the **Delete** key.

5.8. Editing the Create Menu

By default, VR-Forces automatically generates the Create menu based on the entity category and force assignments in the simulation model set. When you change the category or force assignments in the Entity Editor, these changes affect the automatically generated Create menu and the Entity Palette. Inclusion of an entity in these menus is controlled by the Show Object On Create Menu and Object Creation Palette check box. However, if you want to, you can configure a custom Create menu.



This procedure only affects the Create menu. It does not affect the organization of the Entity Palette.

5.8.1. Generating the Create Menu Automatically

To generate the Create menu automatically:

1. Choose Edit → Edit Create Menu. The Menu Editor dialog box opens.
2. Select Automatically Generate Create Menu.
3. Click OK.

5.8.2. Editing the Create Menu Manually

To edit the Create menu manually:

1. Choose Edit → Edit Create Menu. The Menu Editor dialog box opens.
2. Select Use Entity Menu. The window displays a tree list representing the current Create menu.
3. To add a grouping, for example, Entity or Aggregate, or a category:
 - a. Select a current grouping or category.
 - b. Click Add Grouping. The Add Group dialog box opens.
 - c. Select the location for adding the group: as a sibling, above or below the selection, or as a child.
 - d. Click OK. An entry called New Group is added (Figure 5-11).

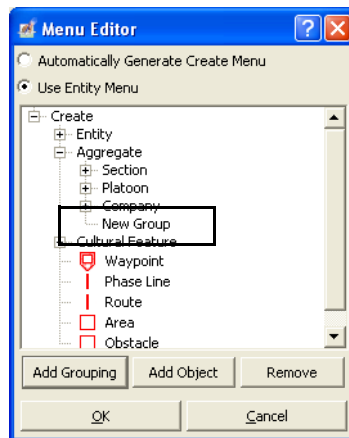


Figure 5-11. Menu Editor with new group added

- e. Double-click New Group to make it editable.
 - f. Type a name for the group.
4. To add an object to the Create menu:
 - a. Select a group title or an object in one of the groups.
 - b. Click Add Object. The Add Object dialog box opens (Figure 5-12).
 - c. Select the location where you want to add the object. If you selected a group title, the options are above or below it, or as a child. If you selected an object, the options are above and below it.
 - d. Select an object to add from the list. Filter the list as necessary to display the categories of objects that you want to add.

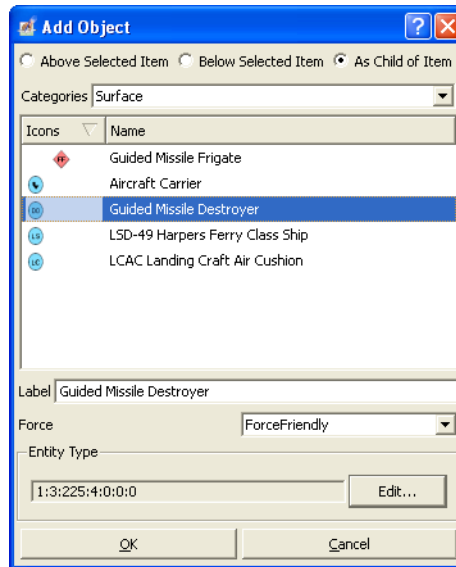


Figure 5-12. Add Object dialog box

- e. Optionally, edit the label and force for the object.
- f. Click OK. The object is added to the menu (Figure 5-13).

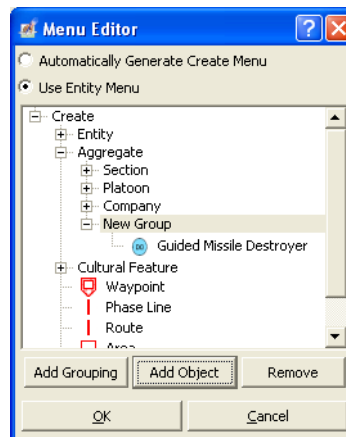


Figure 5-13. Menu Editor with new object added

5. You can edit any of the group or object labels.
6. You can drag an object to another category by dragging it onto the category title.

5.8.3. Undoing (Reverting) Changes to the Create Menu

If you have made changes to the Create menu, but have not saved the simulation model set, you can revert to the original menu.

- To undo changes to the Create menu, choose Edit → Revert Create Menu.

5.9. Editing the Category List

If you want to organize entities into categories that are not included in the default category list, you can add new categories. You can also remove and rename categories.



If you add a category and do not assign it to an entity, it does not get saved when you save the simulation model set.

5.9.1. Adding a New Category

To add a new category:

1. Choose Edit → Edit Categories. The Categories Editor opens (Figure 5-14).

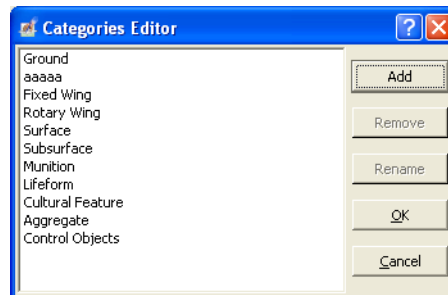


Figure 5-14. Categories Editor

2. Click Add. The New Category dialog box opens.
3. Type the name of the new category.
4. Click OK. The new category is added to the list in the Categories Editor.
5. Click OK.



If you add a category and do not assign it to an entity, it does not get saved when you save the simulation model set.

5.9.2. Removing a Category

To remove a category:

1. Choose Edit → Edit Categories. The Categories Editor opens.
2. Select the category that you want to remove.
3. Click Remove.
4. Click OK.

5.9.3. Renaming a Category

When you rename a category, all entities that are assigned to the old name are re-assigned to the new name. The old name remains in the Categories drop-down list until you save the simulation model set, but it is not listed in the Categories Editor. If you assign an entity to the old category name before saving the SMS, the old name is retained and added to the list in the Categories Editor.

To rename a category:

1. Choose Edit → Edit Categories. The Categories Editor opens.
2. Select the category that you want to rename.
3. Click Rename. The Rename dialog box opens.
4. Type the new name.
5. Click OK. The categories list is updated.
6. Click OK.

5.9.4. Undoing Category Changes

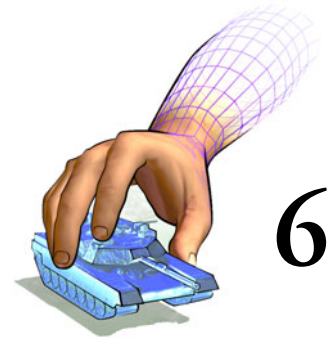
You can undo changes made to the category list and entity category assignments.



When you undo category changes, all changes to the categories list and all changes to entity category 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 category changes:

1. Choose Edit → Revert Categories. A confirmation prompt is displayed.
2. Click Yes.



Advanced Entity Model Editing

This chapter describes how to use the OPD Editor to edit the object parameter database.

The Object Parameter Database Editor	6-2
Starting the OPD Editor	6-3
Loading an Object Parameter Database	6-4
Changing the Object List View	6-4
Editing an Object's Parameters	6-5
Parameter Descriptions	6-5
Editing Individual Parameters	6-6
Editing Parameters for Multiple Objects	6-7
Deleting Parameters and Components	6-7
Restoring Parameters	6-7
Creating a New Object from an Existing Object	6-8
Adding New Components and Parameters	6-8
Adding New Components	6-9
Editing Components	6-9
Adding New Resource Parameters	6-11
Adding New Subordinate Object Parameters	6-11
Adding New Soil Type Parameters	6-12
Editing Variables	6-12
Editing Systems	6-13
Saving Changes to the Object Parameter Database	6-13

6.1. The Object Parameter Database Editor

The Object Parameter Database Editor (OPD Editor) is a graphical user interface for editing the object parameter database. It gives you access to all the components and entity parameters in the object parameter database. Using the OPD Editor can speed up editing of the object parameter database and prevent syntactical errors that can occur when you use a text editor to edit parameters.



If you want to add new entities or customize the entities supplied with VR-Forces, start with the Entity Editor, which is easier to use than the OPD Editor. Use the OPD Editor only if the Entity Editor does not let you make the changes you need to make.



- ♦ When you edit an entity's parameters in the OPD Editor, you are essentially editing the OPE file used by that entity. If more than one entity uses that OPE file, the change will apply to all of the entities.
 - ♦ Do not try to edit an entity model in the Entity Editor and the OPD Editor at the same time. Use one tool, save the changes, and close the tool before opening the other.
-

6.2. Starting the OPD Editor

- To start the OPD Editor, choose Start → Programs → MÄK Technologies → VR-Forces *x.x* → Tools → OPD Editor. The OPD Editor opens ([Figure 6-1](#)).

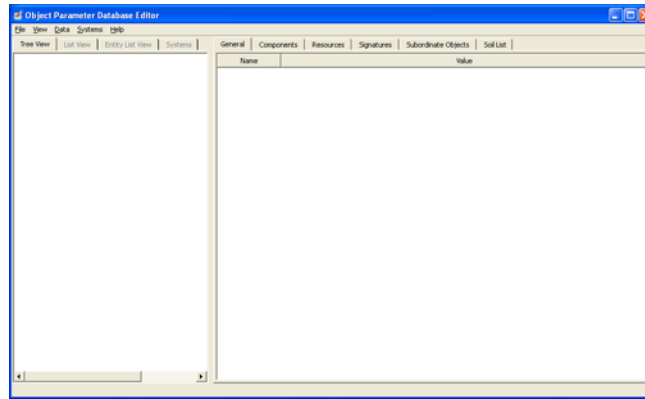


Figure 6-1. OPD Editor

To start the OPD Editor from the Windows or Linux command line:

1. Change directory to *.bin*.
2. Run *vrfOpdEditor*.

6.3. Loading an Object Parameter Database

To edit an object parameter database, you must load it into the editor.

To load an object parameter database:

1. Choose File → Open OPD File. An Open dialog box opens. By default it opens to *./data/simulationModelSets/default*.
2. Select the object parameter database file you want to load, for example, *vrfSim.opd*. You can only load OPD files, not OPE files.
3. Click Open. The file is loaded and is listed as a tree hierarchy in the list window.

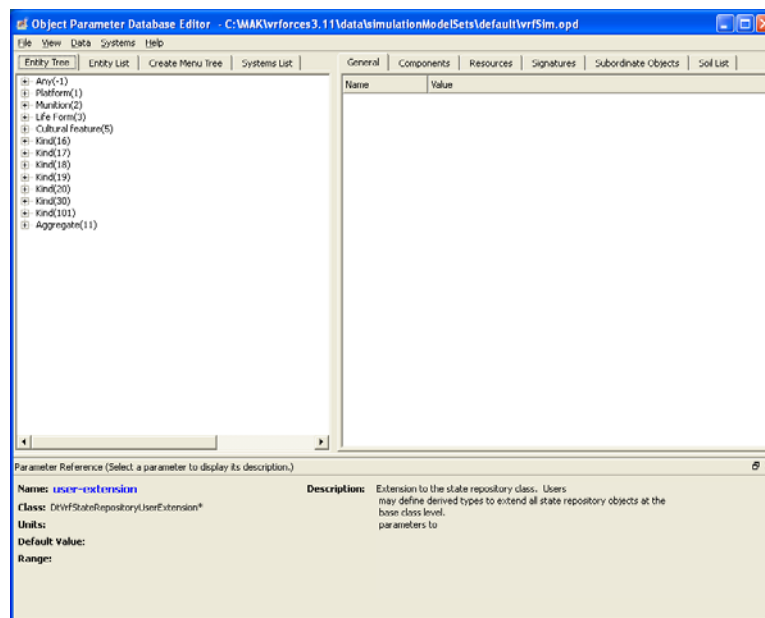


Figure 6-2. OPD Editor with OPD file loaded

6.3.1. Changing the Object List View

The object list has the following views:

- ♦ Entity Tree: The hierarchical tree view allows you to configure any enumeration within the DIS/RPR FOM entity enumeration scheme.
 - ♦ Entity List: The list view lists all objects in the object parameter database.
 - ♦ Create Menu Tree: The entity list view lists entities and objects organized by categories as they are on the Create Menu.
 - ♦ Systems List: The systems view lists the systems configured in systems definition files.
- To change the list view, click the tab for the list you want to view.

6.3.2. Loading a Create Menu Configuration File

When you load an OPD file, the OPD Editor builds the Create Menu Tree view based on the *entity.mst* file for that object parameter database. You can change the Create Menu Tree view by loading any other MST file.

To load a Create Menu configuration file:

1. Choose File → Open Create Menu Configuration File (.mst). An Open dialog box opens. By default it opens to *./data/simulationModelSets/default/gui*.
2. Select the file you want to load, for example, *myentity.mst*.
3. Click Open. The file is loaded and the Create Menu Tree view is updated to match the MST file.

6.4. Editing an Object's Parameters

This section explains how to edit an object's parameters, how to delete parameters, and how to restore parameters that you have changed or deleted. The OPD Editor online help describes each component and component parameter.

6.4.1. Parameter Descriptions

Parameter definitions are documented by component in the online help for the OPD Editor in the Object Parameter Descriptions folder. Individual parameter definitions are displayed in the Parameter Reference panel of the OPD Editor. The definitions are also in the header files and class documentation.

6.4.2. Editing Individual Parameters

The various parameters in the object parameter database are all edited essentially the same way. A parameter's value is a number or string that you type, a value that you select from a list, or a filename that you edit by hand or select from a file chooser. When you edit a parameter, the name of the parameter and all parents is displayed in bold (Figure 6-3).

Name	Value
Radio	
Connections	
Actuators	
Sensors	
Systems	
movement	
system-definition	\$(system-dir)\movement\ground-tracked.sysdef
variable-bindings	
fuel-amount	464
damage	
sensor	
weapon	
weapon-2	
other	
other-2	
Controllers	
org-controller	Priority: 1
process-state-repos...	
process-state-repos...	
tick-period-uses-real...	True
min-tick-period-varia...	-1
min-tick-period	-1
component-type	organization-controller
component-descript...	component-descriptor
set-data	Priority: 2
process-state-repos...	
process-state-repos...	

Figure 6-3. Parameter hierarchy

To edit a parameter:

1. Select the parameter by clicking in the Name or Value column. The Value field becomes editable.
2. Enter the new value. In most cases you enter a new value by typing it. If a parameter value has a File Chooser icon (Figure 6-4), click the icon to open a File Chooser dialog box. If the value has an edit icon, click the icon to open a window that allows you to edit the file.

Systems	
movement	
system-definition	\$(system-dir)\movement\ground-tracked.sysdef
variable-bindings	
fuel-amount	465
damage	
system-definition	\$(system-dir)\damage\ground-heavy-armor.sysdef
sensor	

File Chooser

Edit

Restore

Figure 6-4. Parameter value icons

6.4.3. Editing Parameters for Multiple Objects

To edit parameters for multiple objects:

1. Select the objects that you want to edit by clicking them while holding down the **Ctrl** key. The OPD Editor enters multiple mode. It displays one tab, labeled Multiple. The selected objects do not have to be of the same platform type. If a particular node on the hierarchy has multiple leaf entries, selecting it puts the OPD Editor into multiple edit mode. If there is just one leaf entry, selecting a higher level node leaves the editor in individual edit mode.
2. Edit the parameters on the Multiple tab as you would edit an individual parameter. The changes are applied to all selected objects.

6.4.4. Deleting Parameters and Components

The parameters on the General tab cannot be deleted. Parameters on the other tabs can be deleted.

To delete a parameter or component:

1. Select the parameter or component in the Value list.
2. Choose Data → Delete Parameter Entry or click the Delete icon at the right end of the parameter entry (Figure 6-5).

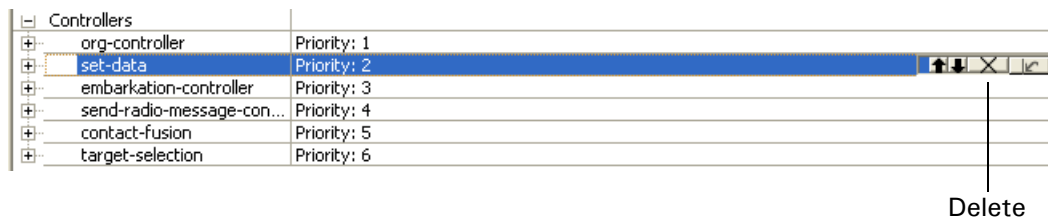


Figure 6-5. Delete parameter icon

6.4.5. Restoring Parameters

If you change a value or delete a parameter or component, you can restore the original value. You can restore at the individual parameter level and at any level in a given hierarchy of entries (Figure 6-3). For example, you can restore a change to the **at-distance** parameter for the **move-to** controller, all edits made to the move-to controller, or all edits made to all Controllers. However, since Actuators are not part of the Controllers hierarchy, restoring a change at the Actuators level would not affect changes to Controllers.

To restore a parameter or component:

1. Select the parameter to restore. If you are restoring a deleted parameter or component, select the parent component.
2. Click Data → Restore Parameter Entry, or click the Restore icon (Figure 6-4).

6.5. Creating a New Object from an Existing Object

You can add new entity types to the object parameter database using an existing object as a template.

To add a new object to the object parameter database:

1. Expand the Entity Tree to the leaf level and select the object on which you want to base the new object. (The other views do not support this procedure.)
2. Choose Data → Create From Existing. The Create Entry dialog box opens (Figure 6-6). It lists the object enumeration for the base object in the Base Entry box.

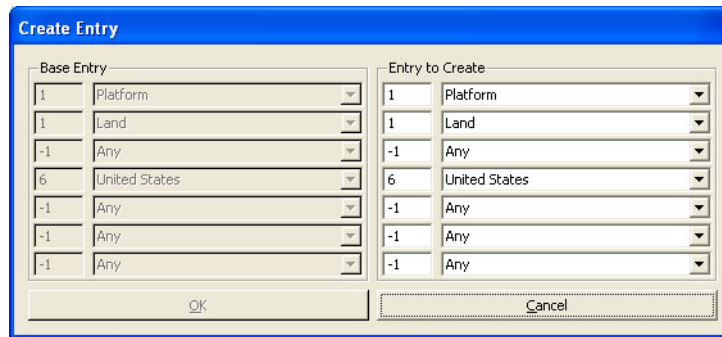


Figure 6-6. Create Entry dialog box

3. In the Entry to Create box, change the enumeration to the one for the object you want to create. You can type the enumeration directly, or build it by selecting options from the lists for each value.
4. Click OK. The new object is added to the Entity Tree in the appropriate location.
5. Edit the parameters for the new object to give it the proper behaviors.
6. Save the object parameter database.

6.6. Adding New Components and Parameters

You cannot add new parameters to the General tab. You can add new components, resources, subordinates, and soil types if an object supports them.



For most entities, the Components tab does not list all of the components that apply to the entity. This is because many components are referenced through the systems that are attached to an entity. If you do not see a component that you are looking for in an entity's component list, look at the component list for the relevant system in the Systems List.

6.6.1. Adding New Components

i

Before you add a component to an entity, consider whether or not the component should be added as part of a new or existing system, rather than being added directly to the entity.

To add a component to an object:

1. Select the Components tab.
2. Select the component type (Actuator, Controller, Sensor) you want to add.
3. Choose Data → Create Parameter, or click the Add icon (Figure 6-7). The Create New Component dialog box opens. The Component list lists all components that are available for this object.

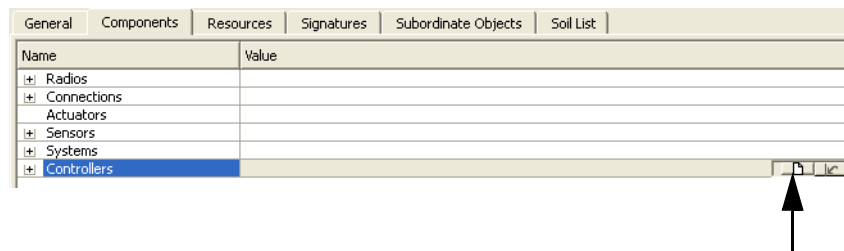


Figure 6-7. Add icon

4. From the Component list, select the component you want to add.
5. Click OK. The component is added to the object.

6.6.2. Editing Components

The Components tab lets you edit actuators, controllers, sensors, and connections. You can edit the individual parameters for components, as described in the other sections of this chapter. You can add and delete components and you can change their priorities.

i

If you do not see a component that you are looking for in an entity's component list, look at the component list for the relevant system in the Systems List.

Changing A Component's Priority

When you change a component's priority, the other components are reprioritized as needed. All affected components are listed in bold.

To change a component's priority:

1. Select the component you want to edit.
2. Click the up or down icon on the value line (Figure 6-8).

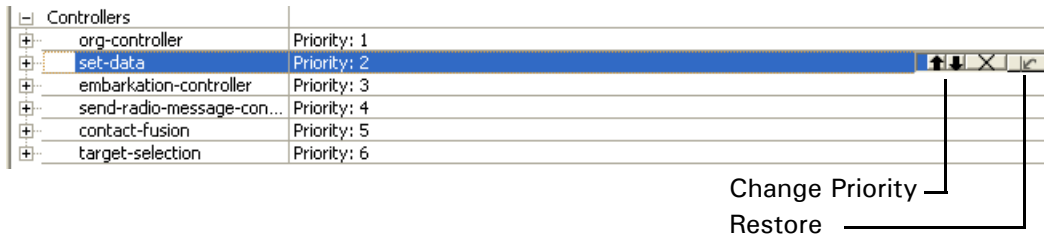


Figure 6-8. Change Priority and Delete Parameter icons

Creating A Connection

When you add a component, you must also add connections between the component and other components it interacts with.

To add a connection:

1. On the Components tab, select Connections.
2. Choose Data → Create Parameter, or click the Add icon (Figure 6-7). The Create Connection dialog box opens (Figure 6-9).

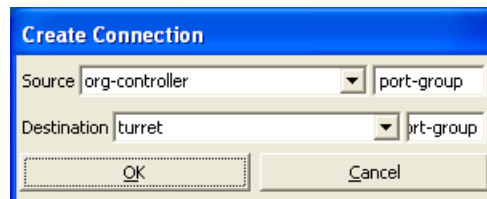


Figure 6-9. Create Connection dialog box

3. Choose a source component from the Source list.
4. Enter the port group for the source component. You must know the name of the port group you need to use. For a list of components and port groups, see the online help.
5. Choose the destination component from the Destination list.
6. Enter the port group for the destination component. You must know the name of the port group you need to use.

7. Click OK. The connection is added to the Connections list.

6.6.3. Adding New Resource Parameters

To add a resource parameter to an object:

1. Select the Resources tab.
2. Select Resources.
3. Choose Data → Create Parameter, or click the Add icon (Figure 6-7). The Create Resource dialog box opens.

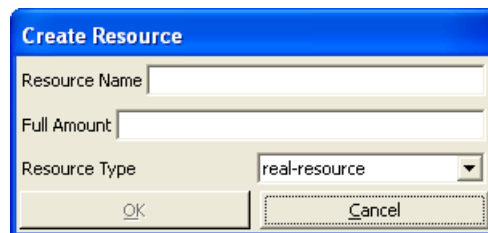


Figure 6-10. Create Resource dialog box

4. In the Resource Name box, enter the name of the resource you want to add. Resource names must start with a letter, not a number. You are responsible for knowing that this is a valid resource supported by your implementation of VR-Forces.
5. In the Full Amount box, enter the amount of the resource to assign to this object.
6. Select a resource type from the Resource Type list.
7. Click OK. The resource is added to the object.

6.6.4. Adding New Subordinate Object Parameters

You are responsible for knowing whether or not a subordinate object is appropriate for a given object.

To add a subordinate object parameter:

1. Select the Subordinate Objects tab.
2. Select Subordinate Objects.
3. Choose Data → Create Parameter, or click the Add icon (Figure 6-7). A subordinate parameter is added.
4. Expand the Subordinate Objects list and edit the parameters for the new subordinate object.

6.6.5. Adding New Soil Type Parameters

To add a soil type parameter to an object:

1. Select the Soil List tab.
2. Expand the Soil List to see which soil types are configured.
3. Choose Data → Create Parameter, or click the Add icon (Figure 6-7). The Add Soil Type dialog box opens (Figure 6-11).

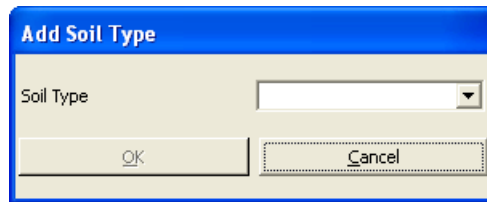


Figure 6-11. Add Soil Type dialog box

4. From the Soil Type list, select the soil type you want to add. The list of soil types is fixed. The Soil Type list contains only the supported soil types that are not already configured for the object.
5. Click OK. The soil type is added to the object.

6.7. Editing Variables

The object parameter database uses variables to specify the location of supporting files, such as ammo selection files, system definition files, and formation files. You can edit these variables.

To edit variable bindings:

1. Choose Data → Edit Variables. The Edit Variables dialog box opens.
2. Change the variable values as desired.
3. Click OK.

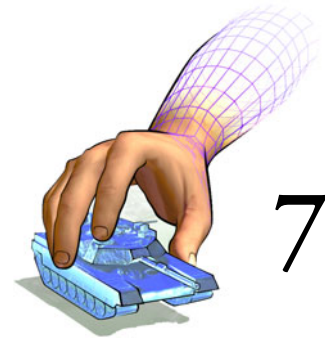
6.8. Editing Systems

Systems are packages of components that get incorporated into entity models by reference. Internally, a system is just a list of components that could just as easily be part of the main OPE file for an entity. Consequently, within any selected system you edit components and parameters exactly the same way that you edit components that are listed for a specific entity. The only difference between a system and other components definitions is that systems contain meta data that enables them to be used in the Entity Editor. You edit a system's metadata in the OPD Editor just as you would any other parameters.

You can create new systems in the OPD Editor, either by copying an existing system or starting a brand new system. These procedures are the same as those described in [“Creating a New Object from an Existing Object,”](#) on page 6-8 and [“Adding New Components and Parameters,”](#) on page 6-8.

6.9. Saving Changes to the Object Parameter Database

- To save an edited object parameter database, choose File → Save OPD file.



The Object Parameter Database

This chapter provides a detailed description of the object parameter database. It will be most useful to VR-Forces users who need to develop and configure new components.

The Object Parameter Database.....	7-2
VR-Forces Classes and Object Parameters	7-6
Object Types	7-6
Parameter Types	7-9
Local and Remote Subcomponents.....	7-9
Component Descriptors and Parameters	7-12
Common Elements of Component Descriptors.....	7-13
Configuring the Priority of Components.....	7-15
Adding a New Entity to the Object Parameter Database	7-16
Systems Definition Files.....	7-18
Creating a System Definition File.....	7-19
Configuring System Connections.....	7-19
Adding Variable Bindings	7-22
Configuring Aggregate Entities	7-23
Adding an Aggregate to the Object Parameters Database.....	7-23
Changing the Echelon ID Category and Echelon ID Level	7-26
Configuring Graphical Objects	7-27
Pathname Interpretation in Simulation Model Set Files	7-27
Updating Object Parameter Database Files for New Releases	7-28

7.1. The Object Parameter Database

VR-Forces entities and objects have many parameters. Parameter values are stored in a set of files 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 an entity and starting entity state values. You can modify the behavior of an entity by changing the parameters for that type of entity. You can add new types of entities 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 before making any changes.

The object parameter database has an object parameter entry (OPE) file for each entity type configured for use by VR-Forces. When you load a scenario, the back-end reads the file *vrjSim.opd*. This file indexes the individual OPE files. When you create an object (either directly or by loading a scenario), VR-Forces reads the parameters for the object from its OPE file. OPE files may reference other files, in particular system definition files.

System definition files group related component and connection specifications into convenient packages that you can reference from OPE files. This makes OPE files easier to read. The system definition files also make it possible to add systems to entities in the Entity Editor.



By default, if successive scenarios use the same object parameter database, VR-Forces does not reload the object parameter database before running each scenario. With the default behavior, if you are making changes to the object parameter database, you must close VR-Forces and restart it to put the changes into effect. However, if you set **forceParameterDbReload** to true, VR-Forces reloads the object parameter database each time you load a scenario.

Example 7.1 is a highly edited example of an OPE file that includes the main features discussed in this chapter and in later chapters. The relevant segments of the example (indicated by arrows) are cross referenced to the sections that discuss them.

Example 7.1

```

1      ;; Entry for M1A2
2      (M1A2
→ 3      (parameter-type "ground-vehicle-param");; "Parameter Types," on
      page 7-9
4      (bounding-geometry-type Volume)
5      (bounding-volume
6      (local-bvol 3.994358 1.929996 0.896801)
7      (offset 0.000000 0.000000 -0.896801)
8      )
9      (formation "Other")
10     (force ForceOther)
→ 11     (category "M1A2");; "Changing the Echelon ID Category and Echelon
      ID Level," on page 7-26
12     (echelon-level "")
13     (is-organized True)
→ 14     (min-tick-period 0.000000);; 15 Hz ;; "Specifying the Tick Rate
      for Components," on page 1-14
15     (min-tick-period-variance 0.000000)
16     (dr-algorithm Rvw)
17     (radios
18     (main-radio
19     (radio-descriptor-type "radio-descriptor")
20     (radio-type "default-radio")
21     (comm-model-name "default-radio-model")
22     (publish-transmitter False)
23     (transmitter-params
24     (radio-type 7 1 225 2 1 20)
25     (initial-frequency 3000000.000000)
26     (power 30.000000)
27     )
28     )
29     )
30     (systems
31     (movement
32     (variable-bindings
33     (DtRwReal
34     (fuel-amount 465.000000
35     )
36     )
37     )
→ 38     (system-definition ;; "Systems Definition Files," on page 7-18
39     (filename "$(system-dir)\movement\ground-tracked.sysdef")
40     )
41     )
42     (sensor
43     (variable-bindings
44     (DtRwVector
45     (sensor-position -1.140000 0.570000 -2.900000
46     )
47     )
48     (DtRwReal
49     (max-range 4000.000000
50     )
51     )
52     )
53     (system-definition
54     (filename "$(system-dir)\sensors\visual-sensor.sysdef")
55     )

```

```

56     )
57     (weapon
58       (variable-bindings
59         (DtRwInt
60           (num-ap-rounds 20
61             )
62           )
63         (DtRwInt
64           (num-heat-rounds 20
65             )
66           )
67         ...
68       )
69       (system-definition
70         (filename "$(system-dir)\weapons/120mm-gun.sysdef")
71       )
72     )
73     (other
74       (system-definition
75         (filename "$(system-dir)\other/spot-report-
76           receiver.sysdef")
77       )
78     )
79     (sensors
80       ...
81     (under-fire-determination-sensor ;; "Configuring Sensors," on
82       page 8-2
83       (component-descriptor-type "under-fire-determination-sensor-
84         descriptor")
85       (component-type "under-fire-determination-sensor")
86       (min-tick-period -1.000000)
87       (min-tick-period-variance -1.000000)
88       (tick-period-uses-real-time True)
89       (process-state-repository-name "")
90       (process-state-repository-type "")
91       (under-fire-time 120.000000)
92       (under-fire-distance 1000.000000)
93       (under-fire-from-friendly True)
94     )
95     (controllers ;; "Component Descriptors and Parameters," on
96       page 7-12
97       ...
98       (set-data
99         (component-descriptor-type "component-descriptor")
100        (component-type "local-entity-set-controller")
101        (min-tick-period -1.000000)
102        (min-tick-period-variance -1.000000)
103        (tick-period-uses-real-time True)
104        (process-state-repository-name "")
105        (process-state-repository-type "")
106      )
107      (embarkation-controller
108        (component-descriptor-type "embarkation-descriptor")
109        (component-type "embarkation-controller")
110        (min-tick-period -1.000000)
111        (min-tick-period-variance -1.000000)
112        (tick-period-uses-real-time True)
113        (process-state-repository-name "")

```

```

112     (process-state-repository-type "")
113     (response-timeout 1.000000)
114     (response-retries 2)
115 )
116 ...
117 )
118 (actuators )
→ 119 (connections ;; "Common Elements of Component Descriptors," on
    page 7-13
120     (connect target-selection:weapon weapon:target-list)
121     (connect target-selection:object-types-to-detect sensor:object-
types-to-detect)
122     (connect sensor:detected-objects contact-fusion:detected-
objects)
123     (connect target-selection:weapon-2 weapon-2:target-list)
124     (connect other:detected-objects contact-fusion:detected-objects)
125 )
126 (state-repository-extension-type "default-vrf-state-repository-
user-extension")
127 ...
→ 128 (max-acceleration 1.270000) ;; Appendix A, Object Parameters
129 (max-deceleration 4.000000)
130 (fuel-efficiency 243.800000)
131 (object-geometry-file
132     (filename ".\objectModel\M1A2_1_1_1_255_1_1_3_-1.gdb")
133 )
134 (soil-list
135     (paved-road
136         (acceleration-factor 1.000000)
137         (stopping-factor 1.000000)
138         (max-speed-factor 1.000000)
139     )
140     (hard-packed
141         (acceleration-factor 0.720000)
142         (stopping-factor 0.720000)
143         (max-speed-factor 0.720000)
144     )
145 )
→ 146 (local-objects ;; "Local and Remote Subcomponents," on page 7-9
147     (state-repository "ground-entity-state-repository")
148     (state-repository-min-tick-period -1.000000)
149     (state-repository-min-tick-period-variance -1.000000)
150     (net-interface "individual-local-entity-net-interface")
→ 151     (net-interface-min-tick-period -1.000000) ;; "Tuning the Network
Interface," on page 1-15
152     (net-interface-min-tick-period-variance -1.000000)
153     (task-manager "local-task-manager")
154     (component-manager "component-manager")
155     (plan-manager "vrfoobject-plan-manager")
156 )
157 (remote-objects
158     ... )
159 (subordinate-objects )
160 (attached-objects )
161 (user-extension )
162 )

```

7.1.1. VR-Forces Classes and Object Parameters

When you create an object in the VR-Forces GUI or by loading a scenario, VR-Forces creates an instance of an object defined by a VR-Forces class. A VR-Forces object class has relationships with other classes that are instantiated along with the object. For example, in the VR-Forces API, each *DtVrfObject* “has a” *DtVrfObjectBaseStateRepository*.

The entries for each object in the object parameter database correspond to VR-Forces classes and tell the simulation engine what type of object to create. For example, if you specify the **parameter-type** ground-vehicle-param” for an entity, VR-Forces knows to use the *DtGroundVehicleParameters* class when it creates the entity.

7.1.2. 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 aggregate objects.

This enumeration scheme describes a tree that goes from a more general specification to a more detailed one. [Figure 7-1](#) illustrates a partial entity type tree. To build an entity 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 entity enumeration scheme, please see the DIS Enumerations Document (*Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications*).

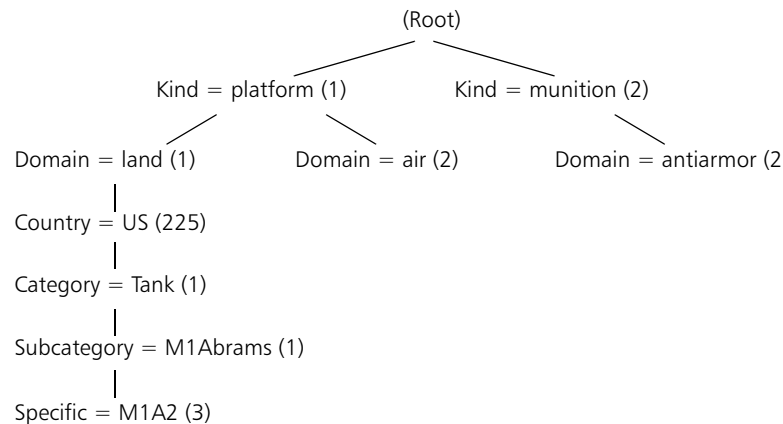


Figure 7-1. Entity type enumeration tree

The Object Super-Type

VR-Forces adds a field to the entity enumeration so that we can manage objects that are not part of the DIS/RPR FOM enumeration scheme, such as aggregate entities and control objects. We enclose the seven-digit enumeration in parentheses and preface it with the object super-type ([Figure 7-2](#)). For example, the entity enumeration for an M1A2, described in the previous paragraph, becomes the object type: 1 (1 1 225 1 1 3 0).

(object-type 1 (1 1 225 1 1 3 0))

|
 object super-type

{
 DIS/RPR FOM

} entity enumeration

Figure 7-2. VR-Forces object-type parameter

[Table 7-1](#) lists the values VR-Forces can use for the object super-type. For more information, please see *objTypeEnum.h* or “[Object Type Enumerations](#),” on page A-9.

Table 7-1: Object super-type field values

Object super-type	Meaning
0	Other
1	Individual (platform entities, individual control objects)
2	Aggregate
3	Pseudo-aggregate
4	True Aggregate

The Object Type Hierarchy

Each simulated object on the Create menu is associated with an **object-type** (for details, please see [“Adding a New Entity or Control Object to vrf.ent,”](#) on page B-5.) When you create an entity, the front-end sends this information to the back-end. The back-end searches through the object parameter database and instantiates the entity with the proper set of values for the **object-type**.

The **object-type** sent by the front-end does not need to exactly match an **object-type** in the object parameter database. Objects in the object parameter database are specified in a hierarchy in which entries at the top 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.)

An object parameter database might have the following entries:

```
(parameter-entry
  (object-type -1 (-1 -1 -1 -1 -1 -1 -1))
  (filename ".\vrfSim\base-sim-object_-1_-1_-1_-1_-1_-1_-1_-1.ope")
)
(parameter-entry
  (object-type 1 (3 1 -1 -1 -1 -1 -1))
  (filename
    ".\vrfSim\generic-lifeform-platform.ope")
)
(parameter-entry
  (object-type 1 (3 1 225 1 1 -1 -1))
  (filename
    ".\vrfSim\friendly-dismounted-infantry.ope")
)
```

As you replace each wildcard value (-1) with a specific enumeration, the scope of the object type narrows. With the illustrated hierarchy, an object with an object type of 1(3 1 225 -1 -1 -1 -1) would have a best match at the generic-lifeform-platform entry. The friendly-dismounted-infantry entry would be too specific for this example enumeration. The order in which parameters are specified in the object parameter database does not affect the hierarchy. The presence of the fixed-wing entity parameter block in this example does not interfere with the hierarchy of the ground-vehicle objects.

7.1.3. 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 7-3](#) illustrates the object parameter type hierarchy.

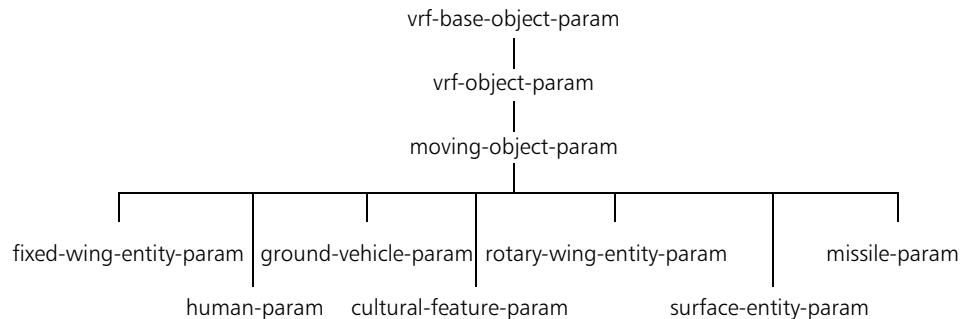


Figure 7-3. Object parameter hierarchy

The **parameter-type** specifies the set of parameters appropriate to an entity, and the entity state repository maintains state information for an entity, so the **parameter-type** and state repository must match. For more information, please see [“The State Repository Hierarchy,”](#) on page 7-11.

7.1.4. Local and Remote Subcomponents

As explained in Section 3.4.2, [“Local Objects and Remote Objects”](#) of *VR-Forces User’s Guide*, each VR-Forces back-end 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 following classes:

- ♦ *DtSimObjectStateRepository*
- ♦ *DtSimObjectNetInterface*
- ♦ *DtSimTaskManager*
- ♦ *DtSimComponentManager*.
- ♦ *DtVrfObjectPlanManager*

These classes are referred to as subcomponents of the object. [Table 7-2](#) maps the subcomponent classes to the parameters that specify them.

Table 7-2: Subcomponent classes and parameters

Class	Parameter
DtSimObjectStateRepository	state-repository
DtSimObjectNetInterface	net-interface

Table 7-2: Subcomponent classes and parameters

Class	Parameter
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 7-3](#) lists the permissible entries for each parameter listed in [Table 7-2](#).

Table 7-3: Permissible subcomponent parameters

Parameter	Permissible values
state-repository	♦ vrf-object-base-state-repository ♦ vrf-overlay-object-state-repository ♦ vrf-object-state-repository ♦ vrf-moving-object-state-repository ♦ vrf-aggregate-state-repository ♦ fixed-wing-entity-state-repository ♦ ground-entity-state-repository ♦ human-state-repository ♦ cultural-feature-state-repository ♦ missile-entity-state-repository ♦ rotary-wing-entity-state-repository ♦ surface-entity-state-repository
net-interface	♦ aggregate-local-entity-net-interface ♦ individual-local-entity-net-interface ♦ local-environmental-net-interface ♦ pseudo-aggregate-local-entity-net-interface ♦ vrf-aggregate-remote-entity-net-interface ♦ vrf-individual-remote-entity-net-interface ♦ vrf-remote-environmental-net-interface ♦ aggregate-remote-entity-net-interface ♦ individual-remote-entity-net-interface ♦ remote-environmental-net-interface
task-manager	♦ local-task-manager
component-manager	♦ component-manager
plan-manager	♦ vrfobject-plan-manager

Every object must have a state repository specified. Most objects need a net interface. Other subcomponents are optional. However, for most entities, you will want to specify a task manager, plan manager, and component manager so that you can control the entity and give it tasks.

Use the existing entries in the object parameter database as guidance if you create a new entity or change an existing entry. A typical entry for local and remote objects is:

```
(local-objects
 (state-repository "ground-entity-state-repository")
 (net-interface "individual-local-entity-net-interface")
 (task-manager "local-task-manager")
 (component-manager "component-manager")
 (plan-manager "vrfoobject-plan-manager")
)
(remote-objects
 (state-repository "vrf-moving-object-state-repository")
 (net-interface "vrf-individual-remote-entity-net-interface")
 (task-manager "")
 (plan-manager "")
 (component-manager ""))
)
```

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, in the object parameter database, each object must specify a state repository within its parameter block and the state repository must match the parameter type for the object.

[Table 7-4](#) lists parameter types and the state repository type required.

Table 7-4: State repository and parameter types

State Repository Type	Parameter Type
ground-entity-state-repository-ground-entity-state-repository	ground-vehicle-param
fixed-wing-entity-state-repository	fixed-wing-entity-param
individual-lifeform-state-repository	individual-lifeform-param
rotary-wing-entity-state-repository	rotary-wing-entity-param
human-state-repository	human-param
cutlural-feature-state-repository	cutltlural-feature-param
missile-entity-state-repository	missile-param
surface-entity-state-repository	surface-entity-param
vrf-moving-object-state-repository	moving-object-param
vrf-overlay-object-state-repository	vrf-object-param
vrf-object-state-repository	

The Net Interface Hierarchy

VR-Forces manages network relationships of local and remote entities 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. Please see Appendix A, [Object Parameters](#) for tables that list the network interfaces used for the entities provided with VR-Forces.

7.2. Component Descriptors and Parameters

The parameters for an entity include a list of all the sensor, controller, and actuator components it uses (please see Section 3.5.4, “[VR-Forces Uses a Component Architecture](#)” in *VR-Forces User’s Guide*.) These components might be in the OPE file or in a system definition file.

A component descriptor is a text block in the object parameter database that describes a component (see Example 7.2). 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. 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.) “[Common Elements of Component Descriptors](#),” on page 7-13 describes the construction of a component descriptor.

Components can receive data from, and convey data to, other components through ports. In an entity’s entry in the object parameter database, you specify which ports a component sends output to and receives data from.

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 component descriptions and a list of port groups, please see the online help for the OPD Editor. For a more detailed discussion of ports and port groups, please see *VR-Forces Developer’s Guide*.

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, please see “[Configuring the Priority of Components](#),” on page 7-15.

7.2.1. 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 entity. It gets used in the connections section of the entity's parameters. Other components may use this name to find the specified component. In Example 7.2, the name of the first component is `move-along` (line 2).
- ♦ **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 7.2, the component descriptor type for the `move-along` component is `ground-move-to-descriptor` (line 3).
- ♦ **component-type** – The component factory creates the type of component specified in this entry. In Example 7.2, the component type is `ground-auto-move-along-controller` (line 4).



The `component-descriptor-type` and `component-type` values are specified in code. You cannot make them up. Please see *compTypes.h* and *compDescTypes.h*.

- ♦ **connections (optional)** – The connections (Example 7.2, line 45) 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 an entity 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 7.2 use this format.

- Specify a port-to-port connection, in which one or both of the ports exists inside of a group, as follows:

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 or the online help for the OPD Editor for the names of port and port groups used by a given component.

All of the entity components that you can use in the object parameter database are described in the online help for the OPD Editor.

Example 7.2 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 7.2

```
1  (controllers
2    (move-along
3      (component-descriptor-type "ground-move-to-descriptor")
4      (component-type "ground-auto-move-along-controller")
5      (min-tick-period -1.000000)
6      (min-tick-period-variance -1.000000)
7      (tick-period-uses-real-time True)
8      (process-state-repository-name "")
9      (process-state-repository-type "")
10     (near-distance 25.000000)
11     (at-distance 1.000000)
12     (approach-speed 2.000000)
13     (stay-in-lane False)
14     (road-center-offset 2.000000)
15   )
16   (patrol-between-points
17     (component-descriptor-type "ground-move-to-descriptor")
18     (component-type "ground-auto-patrol-between-controller")
19     (min-tick-period -1.000000)
20     (min-tick-period-variance -1.000000)
21     (tick-period-uses-real-time True)
22     (process-state-repository-name "")
23     (process-state-repository-type "")
24     (near-distance 25.000000)
25     (at-distance 1.000000)
26     (approach-speed 2.000000)
27     (stay-in-lane False)
28     (road-center-offset 2.000000)
29   )
30 ) ;; end controllers
31 (actuators
32   (powertrain
33     (component-descriptor-type "automotive-component-descriptor")
34     (component-type "automotive-actuator")
```

```
35      (min-tick-period -1.000000)
36      (min-tick-period-variance -1.000000)
37      (tick-period-uses-real-time True)
38      (process-state-repository-name "")
39      (process-state-repository-type "")
40      (art-part-list )
41      (check-terrain-collisions False)
42      (use-high-fidelity-ground-clamping True)
43    )
44  ) ;; end actuators
45  (connections
46    (connect move-along:automotive-control powertrain:automotive-
      control)
47    (connect patrol-between-points:automotive-control
      powertrain:automotive-control)
48  ) ;; end connections
```

The connection in line 46 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.

7.2.2. Configuring the Priority of Components

Each simulation frame, VR-Forces ticks each entity's components in the order in which they are listed in the object parameter database. 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).

For more information about prioritizing components, please see Section 5.3.6, "[Setting the Priority of Components](#)" in *VR-Forces Developer's Guide*.

7.3. Adding a New Entity to the Object Parameter Database

This section explains how to manually add new entries to the object parameter database. The easiest way is to copy the OPE file for a similar entity and change parameters to give it the characteristics you want for the new entity.



The recommended way to add new entities to the object parameters database is to use the Entity Editor (“[Creating a New Entity](#),” on page 5-18) or the OPD Editor (see “[Creating a New Object from an Existing Object](#),” on page 6-8.)



Before making changes to a configuration file, be sure to make a backup copy of the original file.

The object parameter database has an entry for an M1A2 tank, but not an M1A1. If you want to add an M1A1 you could:

1. Copy the *M1A2_1_1_1_225_1_1_3_-1.ope* file and rename it *m1a1.ope*.
2. Open *m1a1.ope*.
3. Change the vehicle name in the very first line of the block.
4. Change the 7-digit entity enumeration portion of the **object-type** to the entity-type for an M1A1. You can look this up in the DIS enumerations document.
5. Change the **category** string to M1A1. (For more information, please see “[Changing the Echelon ID Category and Echelon ID Level](#),” on page 7-26.)
6. Change other parameters as desired and add or delete controllers and actuators to give the new vehicle the characteristics of a M1A1.
7. If you add controllers or actuators, add any necessary connections.
8. Save the *m1a1.ope* file.
9. In the OPD file, add a reference to the new OPE file.

The following code snippet shows the start and end of the parameter block for the M1A2 vehicle in the object parameter database. The minimum changes that you have to make to turn it into an M1A1 (name, object-type, and category) are all in the first few lines of the parameter block.

```
(M1A1
  (parameter-type "ground-vehicle-param")
  (bounding-geometry-type Volume)
  (bounding-volume
    (local-bvol 3.994358 1.929996 0.896801)
    (offset 0.000000 0.000000 -0.896801)
  )
  (formation "Other")
  (force ForceOther)
  (category "M1A1")
  (echelon-level "")
  ...
)
```

The entry in the OPD file would be:

```
(parameter-entry
  (object-type 1 (1 1 225 1 1 2 -1))
  (filename ".\vrfSim\M1A1_1_1_1_225_1_1_2_-1.ope")
)
```

7.4. Systems Definition Files

The Entity Editor allows VR-Forces users to easily add component systems, such as weapons, sensors, and movement, to entities without needing to understand the components, configuration files, and connections required to make these systems work. Editing of systems in the Entity Editor is facilitated by systems definition files (*.sysdef*) in the object parameter database. System definition files are referenced in an entity's OPE file. So, as a byproduct of the use of system definition files, OPE files can be shorter and easier to read. System definition files are read by the OPD Editor and can be edited in it.

Systems 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 an entity. The data could just as easily be written into the OPE file and have the same effect.

The metadata explains to the Entity Editor how to present the system to the user and how to integrate it into the entity model.

Given this information about systems definition files, if you are customizing the object parameter database by adding newly developed components or by using the components provided with VR-Forces, you have the following options:

- ♦ Add the new component configuration directly into OPE files. In this case, the new components will not be accessible from the Entity Editor, but could be edited in the OPD Editor.
- ♦ Package the new component information in a system definition file without metadata and reference the system definition file from the OPE file. Again, the system will not be accessible from the Entity Editor, but can be edited in the OPD Editor.
- ♦ Package the new component information in a system definition file and include metadata. The system will be accessible from the Entity Editor.

7.4.1. Creating a System Definition File

If you want to create a new system definition file, MÄK recommends that you copy an existing file and use it as a template for adding the components that you want to configure into a system. You can use the OPD Editor to do this or a text editor.

The format of a system definition file is as follows:

```
(system-name
  (controllers
  )
  (actuators
  )
  (connections
  )
  (resources
  )
  (meta-data
    (system-name "Spot Report Generator")
    (system-description "Allows the entity to send spot reports though
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, the format is similar to that of the OPE file and other configuration files that use the MTL syntax. 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 list the connection among the components.

7.4.2. Configuring System Connections

Within each system definition file, you list the connections among the components in the file. (For general information about connections, please see [“Common Elements of Component Descriptors,”](#) on page 7-13.) 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, weapons systems must have access to a target list. This section explains how to make these out-of-system connections. (The sample code is from the *M1A2_1_1_1_225_1_1_3_1.ope* file and the *120mm-gun.sysdef* file.)

To understand how system definition files and OPE files are integrated, we will look at:

1. A weapon definition in the OPE file.
2. The related system definition file.
3. The connection section in the OPE file.

The M1A2 is configured with a 120mm gun, using a system definition file.

```
1  (weapon
2    (variable-bindings
3      (DtRwInt
4        (num-ap-rounds 20
5          )
6        )
7      (DtRwInt
8        (gun-art-part-type 4416
9          )
10       )
11      ...
12    (system-definition
13      (filename "$(system-dir)\weapons/120mm-gun.sysdef")
14    )
15  )
```

This block is named `weapon`. It contains several variable bindings. The values attributed to the variables are used in the system definition file. The file itself is specified in the `system-definition` block. For more information, please see [“Adding Variable Bindings,”](#) on page 7-22.

The system definition file has the following relevant code:

```
1  (weapon-120mm-gun
2    (controllers
3      ...
4    )
5    (actuators
6      (main-gun-joy
7        (component-descriptor-type "ballistic-gun-descriptor")
8        ...
9        (process-state-repository-type "")
10       (art-part-list
11         (primary-gun
12           (art-part-type $gun-art-part-type)
13           (parent-part-type $turret-art-part-type)
14           (attach-point 2.331000 -0.125000 0.418000)
15           (art-part-param-list
16             (part-type 13)
17             (part-type 14)
18           )
19         )
20       )
21     ...
22   )
23   (connections
24     (connect system:target-list turret-acquire:target-list)
25     (connect main-gun-control-joy:gun-control main-gun-joy:gun-control)
26     ...
27   )
28   (meta-data
29     ...
30     (parameter-data-list
31       (art-part-parameter-data
32         (parameter-name "gun-art-part-type")
33         (variable-type "DtRwInt")
34         (art-part-category "gun")
```

```
35          (display-label "Gun Part Number")
36      )
37  )
38  )
```

The system component in the first connection (line 24) is a self reference. It links to an equivalent connection in the main OPE file.

The M1A2 OPE entry for weapon creates a variable called gun-art-part-type (line 8). The variable is used in line 12 of the system definition file. Other variables specified in the OPE file are used throughout the system definition file.

The system definition file defines connections for its components. The first connection in the list (line 24) maps system to target-list. If you look at the complete system definition file, you will see that there is no component called system. Nor is there one called target-list.

The use of system in a connection in a system definition file creates a port into the system so that components outside the system can connect to it. Line 24 indicates that a target-list component needs to connect to the weapon system through the system port.

The connections in the M1A2 OPE file are as follows:

```
(connections
  (connect target-selection:weapon weapon:target-list)
  (connect target-selection:object-types-to-detect
    sensor:object-types-to-detect)
  ...
)
```

In the first line in the list of connections, we see that the target-selection component sends output to its weapon output port. (The weapon output port is defined in the target-selection component's block, which is not shown in this section.) The weapon component gets input through its target-list input port. The weapon component is really a reference to the 120mm-gun system. The first connection in the 120mm-gun.sysdef file is:

```
(connect system:target-list turret-acquire:target-list)
```

In this line, "system" is a self-reference that is equivalent to the component name in the OPE file. Therefore, this line could be read as the weapon component sends output (which it received through its target-list input port in the OPE file connection) to its target-list output port. Then the turret-acquire component in the system definition file, takes input through its target-list port. (Figure 7-4) illustrates these connections.

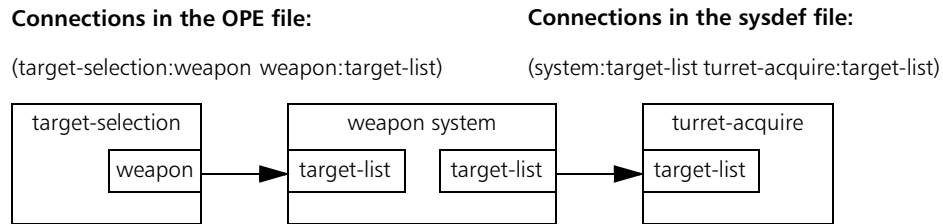


Figure 7-4. Flow of data in component connection

7.4.3. Adding Variable Bindings

Most system definition files are used by multiple entities. However, the values of some parameters in a system definition file may need to vary for the different entities that use it. As previously noted, this is handled by using variables for the values in the system definition and specifying the values for variables in the OPE files. VR-Forces uses variable bindings for the parameters that you are most likely to want to change from entity to entity. If there are other parameters that you want to vary, you can add your own variable bindings.

The following example shows a variable binding for a movement system:

```
(movement
  (variable-bindings
    (DtRwReal
      (fuel-amount 462.000000)
    )
  )
  (system-definition
    (filename "$(system-dir)\movement\ground-tracked.sysdef")
  )
)
```

The variable is used in the SDF as follows:

```
(resources
  (fuel
    (resource-type "real-resource")
    (subresources )
    (amount $fuel-amount)
    (full-amount $fuel-amount)
  )
)
```

As you can see, the variable defined as fuel-amount in the OPE file is referenced as \$fuel-amount in the SDF file.

To add a variable binding to an SDF:

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.
2. In each OPE file that uses this SDF, add a variable binding to the system block that references the SDF you edited. The variable binding must specify the type of the value, the name of the variable, and the value for the variable as shown in the example earlier in this section.



If you do not add a variable binding to an OPE that references the modified SDF, a default value will be used, which may result in unexpected behavior.

7.5. Configuring Aggregate Entities

All aggregate entities have the **parameter-type** “moving-object-param”. Aggregate entities have relatively few individual parameters. The principle function of the various parameter blocks is to specify controllers and formations, and to specify the **echelon-level** for the different levels of aggregates.

7.5.1. Adding an Aggregate to the Object Parameters Database

This section explains how to add a preconfigured aggregate to the object parameter database by directly editing the object parameter database. You can also add new entries with the OPD Editor (see Chapter 6, *Advanced Entity Model Editing*.) The procedure for adding a preconfigured aggregate to the object parameter database is similar to that of adding an entity, except that you must ensure that any subordinate units required by your new aggregate exist in the object parameter database. For example, VR-Forces provides preconfigured squads, platoons, and companies. If you wanted to add a preconfigured brigade, which had several battalions, you would have to add both brigades and battalions to the object parameter database, because the default object parameter database does not have any battalions.

The preconfigured aggregates in the object parameter database are based on the aggregate entries used when you aggregate entities using the Aggregate As command. Although they are not all available through the Aggregate As command, the object parameter database has entries for aggregates up through Corps.

The format for a preconfigured aggregate is:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  ...
  (category "")
  (echelon-level "echelon_level")
  (subordinate-objects
    (sub1
      (object-type 3 (11 1 225 14 3 0 0))
      (position-offset 0.00000 0.000000 0.00000)
```

```
        (heading-offset 0)
      )
      ...
      (subn)
    )
  )
```

where:

- ♦ **parameter-type** must be moving-object-param.
- ♦ **category** should be left blank.
- ♦ **echelon-level** describes the hierarchical unit for this aggregate.
- ♦ The subordinate-objects block lists the entities that you want to be created as subordinates of this aggregate entity. In each subordinate block:
 - **object-type** specifies the object type of an entity defined elsewhere in the object parameter database
 - **position-offset** specifies the location of the entity in aggregate body coordinates
 - **heading-offset** specifies the heading of the entity relative to the aggregate.

Editing a Preconfigured Aggregate

If you want to change the configuration of a preconfigured aggregate or add a variation of a provided type to the object parameter database, edit one of the current object parameter database entries as follows:

1. Copy the aggregate you want to edit to a new OPE file.
2. Change the subordinate list as desired.
3. Change the entity type enumeration to distinguish the entity from other aggregates of its type. The best way to do this is to change the last number in the enumeration.
4. Add a new entry to the *entity.mst* file so that the aggregate will be editable from the Entity Editor and be added to the Entity Palette. If you are using a customized Create menu (as opposed to having it generated automatically), add the new aggregate to *vrf.ent*.

Adding a New Type of Preconfigured Aggregate

This section explains how to add a new type of preconfigured aggregate, such as a battalion, to the object parameter database.

To add a new preconfigured aggregate to the object parameter database:

1. Copy the existing object parameter database entry for the desired hierarchical unit, in this case, the battalion, which looks like:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  ...
  (echelon-level "Bn")
)
```

2. Save the copied file to a new OPE file.

3. Add a category parameter.

```
(category "")
```

4. Add a subordinate-objects block.

5. Add entries for each subordinate entity, for example, four companies. Using the existing preconfigured company as a subordinate, an entry would look like:

```
(sub1
  (object-type 3 (11 1 225 5 -1 -1 1))
  (position-offset 0.00000 0.000000 0.00000)
  (heading-offset 0)
)
```

The resulting entry might look like:

```
(aggregate-local-entity
  (parameter-type "moving-object-param")
  (object-type 3 (11 -1 -1 6 -1 -1 -1))
  (category "")
  (echelon-level "Bn")
  (subordinate-objects
    (sub1
      (object-type 3 (11 1 225 5 -1 -1 1))
      (position-offset 0.00000 0.000000 0.00000)
      (heading-offset 0)
    )
    (sub2
      (object-type 3 (11 1 225 5 -1 -1 1))
      (position-offset 1000.00000 0.000000 0.00000)
      (heading-offset 0)
    )
    ...
  )
)
```

6. Add a new entry to the *vrSim.opd* file with an appropriate object type value.

7. Add a new entry to the *entity.mst* file so that the aggregate will be editable from the Entity Editor and be added to the Entity Palette. If you are using a customized Create menu (as opposed to having it generated automatically), add the new aggregate to *vrf.ent*.

7.6. Changing the Echelon ID Category and Echelon ID Level

You can change the category and echelon ID level used in echelon IDs by editing the object parameter database. Changes to the object parameter database take effect the next time you start VR-Forces.



Before making changes to a configuration file, be sure to make a backup copy of the original file.

To change the category or echelon name used for a platform, such as a tank:

1. Open the OPE file for the entity in a text editor.
2. Edit the value for the **category** or **echelon-level** parameter.
3. Save the file.



You can also add new entries with the OPD Editor. For details, please see Chapter 6, [Advanced Entity Model Editing](#).



- ♦ Entity names have the following restrictions on their length:
 - Entities – 11 characters
 - Aggregates – 10 characters in HLA exercises, 32 characters in DIS.

Since the category is used to build the default entity name, make sure that it is short enough, in combination with the incremental entity number (for example, M1A2 23), that it will not exceed the character limits for entity names. If the default entity name is too long, it gets truncated and will not work properly.

- ♦ If you change the **category** or **echelon-level** parameters (or both) for any entry, existing scenarios that reference this type of object will not work unless you edit the order of battle file, object map file, and plan file so that the object names use the new **category** and **echelon-level** strings.
-

7.7. Configuring Graphical Objects

The object parameter database includes parameters for graphical objects. Graphical objects 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 graphical objects. You would have to include parameters for them in the object parameter database.

7.8. Pathname Interpretation in Simulation Model Set Files

The files that make up a simulation model set, such as the SMS file, the OPD file, and OPE 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 can 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. The following lines from the *default.sms* file show a binding (`model-set-directory`) and its use:

```
(simulation-model-set
  (variable-bindings
    (DtRwString
      (model-set-directory "default")
    )
  )
  (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 OPE files, as shown.

From *vrfSim.opd*:

```
(variable-bindings
  (DtRwString
    (system-dir "../data/simulationModelSets/default/vrfSim/systems")
  )
  (DtRwString
    (ammoselect-dir
      "../data/simulationModelSets/default/vrfSim/ammoselect")
    )
  (DtRwString
    (damage-dir "../data/simulationModelSets/default/vrfSim/damage")
  )
  ...
)
```

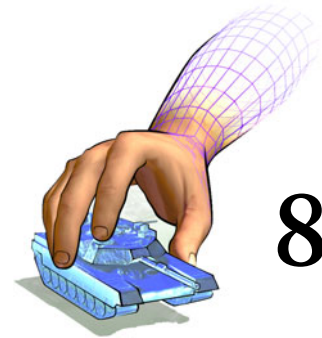
From *BMP2_1_1_1_222_2_2_1_1.ope* (system-dir variable):

```
(systems
  (tracked-movement
    (variable-bindings
      (DtRwReal
        (fuel-amount 462.000000)
      )
    )
  )
  (system-definition
    (filename "$(system-dir)\tracked-movement.sysdef")
  )
  ...
)
```

7.9. Updating Object Parameter Database Files for New Releases

New releases of VR-Forces frequently include new components and new parameters for existing components. If you are using a customized object parameter database you will probably want to update it to take advantage of the new components and parameters. Unfortunately, there is no automatic way to migrate an object parameter database. We recommend using one or a combination of the following methods:

- ♦ Start with the object parameter database for the latest release and add your customizations to it.
- ♦ Use a text comparison tool (diff tool) to merge your customized files with the release version of the object parameter database.
- ♦ To merge in new parameters for existing components, in the OPD Editor, select the specific entities that you have customized, then save the object parameter database. The new parameters will be added with default values.
- ♦ The only way to add new components to an object parameter database or replace outmoded components (for example, to replace the target detection sensor with the new sensor mechanism in an older object parameter database) is to add them by hand in individual files or with the OPD Editor.
- ♦ Use multiple simulation model sets. VR-Forces can load multiple simulation model sets (for details, please see [“Specifying Multiple Simulation Model Sets,”](#) on page 5-6.) If your customizations do not overlap the default SMS or are relatively easy to separate from the default SMS, you can create a separate SMS and load it along with the SMS supplied with VR-Forces.



Configuring Object Behaviors

This chapter describes configuration files that affect entity behavior and the effects of munitions.

Configuring Sensors.....	8-2
Configuring a Sensor in an OPE File.....	8-2
Editing a Sensor System Definition File.....	8-3
Specifying a Sensor Signature	8-4
Adding a New Sensor Type.....	8-4
Probability and Selection Tables for Munitions	8-7
Hit Probability Tables.....	8-7
Damage Probability Tables	8-9
Detection Probability Tables.....	8-11
Ammo Select Tables	8-12
Configuring an Entity to Take Damage from a Munition.....	8-13
Configuring Formations.....	8-13
Formation Files	8-14
Creating a User-Defined Formation	8-15
Configuring Emitters.....	8-16
Configuring Radios.....	8-18
Configuring Transmitters	8-18
Configuring Joysticks.....	8-19
Configuring Obstruction Avoidance	8-20
Configuring Object Types	8-20
Configuring Vector Feature Types	8-21
Configuring Embarkation Ingress and Egress Points	8-22

8.1. Configuring Sensors

VR-Forces allows you to easily add different types of sensors to entities. VR-Forces includes several sensor systems and you can easily add them to entities using the Entity Editor or the OPD Editor. This section provides additional details about how sensors are configured in the OPE file and system definition file. A complete sensor configuration requires the following:

- One or more system definition entries in the sensor block in the sensing entity's OPE file
- An entry in the connections section of the entity's OPE file
- A sensor signature in the OPE file for each entity that can be sensed by a sensor type
- An signature propagator entry in the *physicalWorldParam.mtl* file (one per sensor type.)

VR-Forces includes radar, infrared, visual, and sonar sensors. The entities 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 Section 5.10, “Sensors”, in *VR-Forces Developer's Guide*.

8.1.1. Configuring a Sensor in an OPE File

All sensors provided with VR-Forces are signature-object-sensors. They are distinguished by the domain in which they operate (visual, radar, infrared, or sonar.)

To add a sensor to an entity:

1. Copy an existing sensor block in the OPE file and paste it into the sensors section.
2. Change the name of the system definition file to point to the system you want to add.
3. Change the values for the variables that must be specified for the system.

Connecting a Sensor

If you add a 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 following is from the connections block for an M1A2. The sensor gets input through its object-types-to-detect input port. It sends that data out through its detected-objects output port:

```
(connect target-selection:object-types-to-detect sensor:object-types-to-detect)
(connect sensor:detected-objects contact-fusion:detected-objects)
```

If you were to add, for example, an infrared sensor to an entity and you called the sensor block infrared-sensor, you would add the following line:

```
(connect infrared-sensor:detected-objects target-selection:detected-objects)
```

8.1.2. Editing a Sensor System Definition File

The preferred method for editing a sensor system definition file, as with all system definition files, is to edit it in the OPD Editor. You can also edit the file by hand. The following example is from the visual sensor system definition file:

```
(visual-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 "visual")
  (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-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)
)
```

As you can see from viewing the file, most of the parameters that you would want to vary from entity to entity are variables that are set in the variable binding section of the OPE file.

8.1.3. Specifying a Sensor Signature

A sensor signature indicates that an entity can be detected by other entities that have a sensor for that signature. The value of the signature using the standard VR-Forces propagator represents the maximum distance, in kilometers, at which the entity can be detected. However, this value is affected by modifiers and the calculations made by the propagator.

The modifiers specify multipliers for that base value based on the current state of the object. For example, an entity 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 will be 7.5 when moving and 5.0 when not moving.

The following is the sensor signature block from an M1A2:

```
(sensor-signatures
  (visual
    (base-signature 4.000000)
    (modifiers )
  )
  (radar
    (base-signature 27.000000)
    (modifiers )
  )
  (infrared
    (base-signature 4.000000)
    (modifiers )
  )
)
```

This sensor-signature block shows that an M1A2 can be detected by visual, infrared, and radar sensors. If you do not want an entity to be detectable by a particular type of sensor, remove it from the OPE file. If you want it to be detectable by an additional sensor type, add it to the OPE file. You can edit the values of sensor signatures in the Entity Editor. You cannot add sensor signatures in the Entity Editor. You can edit the values of sensor-signatures and add new sensor signatures in the OPD Editor.

8.1.4. Adding a New Sensor Type

You can easily add new sensor types to your entities. Suppose that you want to add an acoustic sensor. You would do the following:

1. Add a block to the signature-propagator-params block in *physicalWorldParams.mtl*.
2. Add a sensor block to each entity that can detect targets acoustically and connect the sensor to the detected-objects port.
3. Add a sensor signature to each entity 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. The following is a portion of this 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 "standard-propagator")
      (check-line-of-sight True)
    )
  )
)
```

If you edit the file to add an acoustic sensor, it might look like the following:

```
(physical-world-params
  (signature-propagator-params
    (visual
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight True)
    )
    (acoustic
      (propagator-descriptor-type "standard-propagator-descriptor")
      (propagator-type "standard-propagator")
      (check-line-of-sight False)
    )
    ...
  )
)
```

You don't have to be able to see an entity 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 [“Configuring a Sensor in an OPE File,”](#) on page 8-2. A new acoustic sensor might look like the following:

```
(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)
)
```

The connection for the acoustic sensor would look like:

```
(connect target-selection:object-types-to-detect sensor:object-types-to-detect)
(connect sensor:detected-objects contact-fusion:detected-objects)
```


Adding a Sensor Signature

If you add a new sensor, you need to add a signature for it to each entity that can be detected by the sensor. If you add an acoustic sensor, a sensor signature might look like the following:

```
(sensor-signatures
  (visual
    (base-signature 4.000000)
    (modifiers )
  )
  (acoustic
    (base-signature 2.000000)
    (modifiers )
  )
  ...
)
```

You can probably see entities at a greater distance than you can hear them, so we have a lower base-signature value.

8.2. Probability and Selection Tables for Munitions

When the entities in a scenario shoot at each other, VR-Forces uses an ammo select table to select the ammunition to be used. It uses a set of tables to calculate the probability of:

- ♦ Hitting a target (hit probability table)
- ♦ Detecting targets (detection probability tables)
- ♦ Being damaged when hit by hostile munitions (damage probability table).

8.2.1. Hit Probability Tables

A weapon system definition file specifies a hit probability file. This file contains a hit probability table that is used by the system's actuator to determine whether or not 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”.

i

- ♦ 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:

$$(- 5.00000E-008 * r^2) + (-300000E-012 * r) + 1.0$$

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:

$$(- 5.00000E-008 * r^2) + (0.0 * r) + 1.0$$

The following is an example of the *M1A2MainGun.hit* hit probability table:

```
(hit-probability-table
  (entity-range
    (entity-type 1 1 -1 -1 -1 -1 -1)
    (range-determinant
      (range-list
        (range 1000.000000
          (probability 0.90000)
        )
        (range 2000.000000
          (probability 0.8000)
        )
        (range 3000.000000
          (probability 0.75000)
        )
        (range 4000.000000
          (probability 0.70000)
        )
      )
    )
  )
)
```

8.2.2. Damage Probability Tables

The damage actuator uses the damage probability table specified by its component descriptor to determine direct and indirect fire damage based on several factors. The 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.

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), and a **range-determinant**. The entry is intended for use with angles of incidence that go from the previous entry's specified angle to the current one (or in the case of the first entry, from 0.) The range determinant is exactly the same as the range determinant described in [“Hit Probability Tables,”](#) on page 8-7. The damage actuator provided by MÄK, looks for a probability entry with the name **catastrophic-kill**.



The range for missiles is the range from the missile to the target, not from the entity that fired the missile.

The following damage probability table fragment has two surface entries, front and rear. The first uses a coefficients range determinant, the second surface entry uses a **range-list** determinant.

```
(damage-table
  (front
    (angle-of-incidence
      (angle 0.5236) ;; 30 degrees
      (range-determinant
        (coefficients
          (catastrophic-kill -5.0000E-008 0.0000 1.0000)
        )
      )
    )
    (angle-of-incidence
      (angle 1.0472) ;; 60 degrees
      (range-determinant
        (coefficients
          (catastrophic-kill -5.0000E-008 0.0000 1.0000)
        )
      )
    )
  )
  (rear
    (angle-of-incidence
      (angle 0.523600)
      (range-determinant
        (range-list
          (range 2000.000000
            (catastrophic-kill 0.800000)
          )
        )
      )
    )
  )
)
```

```
        (range 3000.000000
          (catastrophic-kill 0.600000)
        )
        (range 4000.000000
          (catastrophic-kill 0.300000)
        )
      )
    )
  (angle-of-incidence
    (angle 1.047200)
    (range-determinant
      (range-list
        (range 2000.000000
          (catastrophic-kill 0.800000)
        )
        (range 3000.000000
          (catastrophic-kill 0.600000)
        )
        (range 4000.000000
          (catastrophic-kill 0.300000)
        )
      )
    )
  )
)
```

8.2.3. Detection Probability Tables

Detection probability tables are comma-delimited files that are most easily read in a spreadsheet. As illustrated in [Figure 8-1](#), each column represents a detection interval, in seconds, from 0 through 160. The rows represent the apparent signature from one through 4. The detection probability is determined by matching the apparent signature to the detection interval used. The detection probability is used in the Detect condition in plans. The apparent signature maps to the certainty levels as follows:

- ♦ 1 – Detected
- ♦ 2 – Classified
- ♦ 3 – Identified
- ♦ 4 – Full Knowledge.

	A	B	C	D	E	F	G	H	I	J	K
1	CID Table-radar Version 1.0		Detection interval in seconds >>								
2			0	1	5	10	20	40	80	160	
3	Apparent signature	1	1	1	2	2	2	3	3	3	
4		1.5	1	1	2	2	3	3	3	3	
5		2	1	1	2	2	3	3	3	4	
6		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	
9											
10											

Figure 8-1. Detection table displayed in a spreadsheet

For details about CID detection levels, please see Section 8.6, “[Target Detection](#)”, in *VR-Forces User’s Guide*.

8.2.4. Ammo Select Tables

The ballistic gun controller descriptor refers to an ammo select table file, which contains the ammo select table for that controller. Each entry in this file specifies a target type (using the best-match method) and a prioritized list of ammunition that should be used for that type of target (highest priority listed first). The name of the entry is not used, and so may be something descriptive for readability. Each entry has the following fields:

- ♦ **target-type** – a 7 digit entity type enumeration, which can use -1s to denote wild-cards.
- ♦ **ammo** – list of ammunition entries. The names of the ammunition entries are assumed to be the resource names. If the resource name is found in the Object Resource Manager, the amount of the ammunition is decremented appropriately, and is not used if the resource amount is 0. In this case, the next ammunition entry (if any) in the ammunition list is used. Each ammunition entry has the following fields:
 - **munition-type** – 7 digit entity type enumeration. This is the entity type that is used in the fire and detonation interactions created, and should NOT contain -1s.
 - **warhead** – VR-Link warhead enumeration value to use in fire and detonation interactions for this munition.

The following is an example of an Ammo Select Table:

```
(ammo-select-table
  (default ;; matches any ground vehicle
    (target-type 1 1 -1 -1 -1 -1 -1)
    (ammo
      (M830-HEAT-120mm ;; these names need to match resource names
        (munition-type 2 2 225 2 13 3 0)
        (warhead 0)
      )
      (M829A1-AP-120mm ;; these names need to match resource names
        (munition-type 2 2 225 2 13 2 0)
        (warhead 0)
      )
    )
  )
  (battle-tank ;; matches any tank
    (target-type 1 1 -1 1 -1 -1 -1)
    (ammo
      (M829A1-AP-120mm ;; these names need to match resource names
        (munition-type 2 2 225 2 13 2 0)
        (warhead 0)
      )
      (M830-HEAT-120mm ;; these names need to match resource names
        (munition-type 2 2 225 2 13 3 0)
        (warhead 0)
      )
    )
  )
)
```

8.2.5. Configuring an Entity to Take Damage from a Munition

If you want an entity to take damage from a particular type of munition, it must have an ammo-entry block for that type of munition in its damage-component. For example, the individual lifeform damage actuator is not configured with an entry for M107-155mm, which means it would not take damage from artillery. If you want to make individual lifeforms vulnerable to this type of munition, you must add an entry for this munition type to the entity's damage model.

8.3. Configuring Formations

The object parameter database lists the formations available to an aggregate entity in the **aggregate-formation-controller** block in its OPE file. Each formation is identified by its name and the relative path to the formation file, for example:

```
(aggregate-formation-controller
  (component-descriptor-type "formation-controller-descriptor")
  (component-type "aggregate-formation-controller")
  ...
  (formation-list
    (column-formation
      (formation Column)
      (formation-file
        (filename ".\formation\platoonColumn.frm")
      )
    )
    (line-formation
      (formation Line)
      (formation-file
        (filename ".\formation\platoonLine.fm")
      )
    )
  )
  ...
)
```

8.3.1. Formation Files

A formation file specifies the name of the formation and has an entry for each entity 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 entities are promoted to aggregate leader if the leader is incapacitated.
- The **leader-promotion-id** specifies the aggregate 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 entity follows the entity in front of it by 25 meters. Therefore, the **leader-promotion-id** for each entry is the **promotion-id** of the entity 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 entities into an aggregate 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 entity 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 entities.

If you want to change the layout of the formations provided with VR-Forces, you can edit the formation files.

8.3.2. Creating a User-Defined Formation

You can create formations other than those provided by VR-Forces. A user-defined formation must have the name `UserFormation $x$` , where x is a number from one to as many formations as you define.

To create a formation:

1. Using one of the user formations 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 OPE file for each aggregate that you want to be able to use the formation. Be sure to use the same structure and naming conventions as the `user-formation-1` entry. For example:

```
(user-formation-2
  (formation UserFormation2)
  (formation-file
    (filename ".\formation\wingformation.frm")
  )
)
```

- To specify a user-defined formation in the VR-Forces GUI, enter the formation name in the Formation field of the Set Formation dialog box. User-defined formations are not included in the list of formations.

We recommend that you configure your formations so that entities 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.

8.4. Configuring Emitters

All fixed-wing aircraft have emitters as part of their radar sensor. 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 is as follows:

```
(active-radar-sensor-system
  (systems )
  (sensors
    (radar-sensor
      (component-descriptor-type "radar-signature-sensor-descriptor")
      (component-type "radar-signature-sensor")
      . . .
      (emitter-system
        (system-id 0)
        (system-name-enum 5570)
        (system-function-enum 5)
        (beam-list
          (beam-0
            (index 0)
            (type "track")
            (frequency 4999999488.000000)
            (frequency-range 5000.000000)
            (power 70.000000)
            (pulse-repetition-frequency 0.000000)
            (pulse-width 0.000000)
            (azimuth-center 0.000000)
            (azimuth-sweep 3.142857)
            (elevation-center 0.000000)
            (elevation-sweep 1.000000)
            (sweep-sync 0.000000)
            (beam-function-enum 5)
            (beam-param-index 0)
          )
        )
      )
    )
  )
)
```

To configure multiple emitters for an entity, add additional radar-sensor blocks. Each block must have a unique name and each **system-id** parameter must be unique. For example:

```
(radar-sensor-1
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type "radar-signature-sensor")
  . . .
  (emitter-system
    (system-id 0)
    ...
  )
)
(radar-sensor-2
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type "radar-signature-sensor")
  . . .
  (emitter-system
    (system-id 1)
    ...
  )
)
```

To add multiple beams on an emitter, add additional beam-list blocks and change the name of each block. For example:

```
(radar-sensor
  (component-descriptor-type "radar-signature-sensor-descriptor")
  (component-type "radar-signature-sensor")
  . . .
  (emitter-system
    (system-id 0)
    (system-name-enum 5570)
    (system-function-enum 5)
    (beam-list
      (beam-0
        ...
      )
      (beam-1
        ...
      )
    )
  )
)
```

8.5. Configuring Radios

All radios are *DtVrfRadio* classes or subclasses. The radio is responsible for managing incoming and outgoing radio messages on a single entity. Each radio is associated with a specific communication model, and uses this communication model to send and receive messages.

Radios are configured in an entity's OPE file. The radios block in the OPE file lists all the radios on the entity, 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 radio can also specify transmitter parameters (please see the next section for details.)

A typical OPE radio entry is as follows:

```
(main-radio
  (radio-descriptor-type "radio-descriptor")
  (radio-type "default-radio")
  (comm-model-name "default-radio-model")
  (publish-transmitter False)
  (transmitter-params
    (radio-type 7 1 225 2 1 20)
    (initial-frequency 3000000.000000)
    (power 30.000000)
  )
)
```

8.5.1. Configuring Transmitters

You can configure entities to publish a transmitter object corresponding to a radio. A transmitter object is a DIS or HLA object that represents an object that is capable of transmitting radio messages. By default, radio transmitters are not published. You can enable publication of transmitters in the OPE file.

To enable publication of transmitters, in an entity's radio entry, set the **publish-transmitter** parameter to True for each radio that is to publish its transmitter state.

You can specify some of the fields in the transmitter state that will be published to the network. These parameters have no effect on the operation of the radio. They only affect which values are published that relate to this transmitter. Some external communication effects servers require that the transmitter state be published for the server to operate, and may use some of the published parameters.

8.6. Configuring Joysticks

The `./config/joyParam.dat` file specifies the default values for joystick parameters. The parameters are:

- ♦ **joy-device-type** – Possible values are:
 - `direct-input-joydevice` – used for direct-input joysticks on Windows 2000 or XP
 - `pre-direct-input-joydevice` – used on Windows NT, for joysticks that do not use DirectInput.
 - `linux-joydevice` – used for joysticks on Linux.

Default: `direct-input-joydevice`.

- ♦ **joy-trans-sensitivity** and **joy-rot-sensitivity** – affect the translation and rotation displacement required to make VR-Forces respond. The value should be between 0 and 1.0, with 0 being not sensitive at all, and 1.0 being the most sensitive it can be.
- ♦ **joy-trans-gain** and **joy-rot-gain** – affect the velocity of motion for a given displacement. The value is a multiplier, so the default value of 1.0 has no effect on the gain.
- ♦ **joy-trans-invert-x** (y, z) and **joy-rot-invert-x** (y, z) – allow you to invert axes on an individual basis.



The use of joysticks is supported only when you are running in combined mode.

You can override the default values for joystick parameters for specific entity types in the object parameter database. Change the values in the joystick parameter block for the system definition file.

```
(joystick
  (component-descriptor-type "base-joy-controller-descriptor")
  (component-type "ground-move-joystick-controller")
  (joy-trans-sensitivity 0.3)
  (joy-rot-sensitivity 0.3)
  (joy-trans-gain 0.1)
  (joy-rot-gain 0.2)
)
(turret-joystick
  (component-descriptor-type "base-joy-controller-descriptor")
  (component-type "ground-turret-joystick-controller")
  (joy-trans-sensitivity 0.8)
  (joy-rot-sensitivity 0.8)
  (joy-trans-gain 0.1)
  (joy-rot-gain 0.2)
)
)
```

The rotary wing controller has some extra values that you can set.

```
(joystick
  (component-descriptor-type "rotary-wing-joy-controller-descriptor")
  (component-type "rotary-wing-joy-flight-controller")

  ;; controller rise time. Default: 1.5
  ;; rotary-wing-joy-controller-descriptor only
  (controller-rise-time 1.6)

  ;; pitch feedback constant. Default: 1.0
  ;; rotary-wing-joy-controller-descriptor only
  (pitch-control-feedback-constant 1.0)

  ;; roll feedback constant. Default: 2.2
  ;; rotary-wing-joy-controller-descriptor only
  (roll-control-feedback-constant 2.265)

  ;; yaw feedback constant. Default: 2
  ;; rotary-wing-joy-controller-descriptor only
  (yaw-control-feedback-constant 2.1)
)
```

8.7. 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.

8.7.1. 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, the following 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
)
```

8.7.2. Configuring Vector Feature Types

Configuring avoidance of vector features relies on feature specifications in GDB databases. To configure a vector feature type for obstruction avoidance, modify the appropriate **vector-type** parameter in the **vector-obstruction-type** parameter block for the obstruction sensor in an entity's object parameter database entry.

The vector obstruction types are used to create *DtSpecifications*, which are then used by the vector network intersection metrics in determining if obstructions are the requisite types or not. Any attribute specified for avoidance must be in the vector feature for it to be considered an obstruction. Because the vector feature may contain additional properties, simple vector obstruction types can map to a large number of features.

For example, to specify all types of buildings, add a **vector-type** parameter with the single attribute:

```
(vector-obstruction-type
  (vector-type
    (string-attributes
      (DtFeatureGroup "MAK000")
    )
  )
)
```



For details about the MÄK specification model, please see Chapter A, *The MÄK Specification Model* in *MÄK Terrain Database Tool User's Guide*.

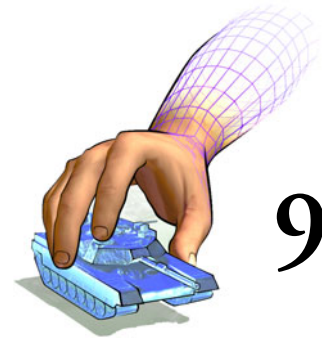
If you only want to avoid a specific feature, create a more detailed vector-type. For example, the following will cause only standard lights to be considered obstructions.

```
(vector-obstruction-type
;; Light Standard (street light) detailed spec
;; illustrates how to create a detailed obstruction type
(vector-type
  (integer-attributes
    (DtFidCode 537)
    (DtFeatureType 0)
    (DtSmc 1)
    (DtSedrisCodes 0)
    (DtOrientation 0)
    (DtHeight 3)
  )
  (string-attributes
    (DtFeatureName "537 Light Standard")
    (DtFeatureGroupName "Power")
    (DtFeatureGroup "MAK050")
    (DtDescription "US-Light Standard/Light Support UK-Light
      Standard/Light Support/Lamp Post")
    (ECC "AL110")
  )
  (real-attributes
    (GDB_Width 0.0)
    (GDB_Length 2.0)
  )
)
))
```

8.8. Configuring Embarkation Ingress and Egress Points

Entities that are configured to allow other entities to embark on them have an occupancy-director-controller component. The locations at which entities can embark on an entity are the ingress points. The locations from which entities can disembark are egress points. The number of entities that can embark on an entity is defined by the embarkation-slots parameter.

You can add to or edit the ingress and egress points for an entity in the OPD Editor. If you need to edit these values, please contact MÄK Technologies support for assistance.



Using VR-Forces with an External Communications Effects Server

This chapter explains how to configure VR-Forces to interoperate with an external communications effects server.

Introduction	9-2
Configuring VR-Forces.....	9-2
Enabling Radio Transmitter Publishing	9-3
Enabling the External Communications Model	9-3
Configuring the External Communications Model.....	9-4
Using QualNet.....	9-4
Running VR-Forces with QualNet	9-5
Example Scenario.....	9-5
Running the Scenario	9-6

9.1. Introduction

You can configure VR-Forces to use an external communications effects server to control the transmission of radio messages between VR-Forces entities. The communications effects server determines which entities are able to communicate with each other, and how long it takes a message to be transmitted from the originator of the message to the receiver of the message.

VR-Forces supports the interface specified in the file *./doc/QualNet_HLA_interface_RPR-FOM 1.0_extensions_ICD3.pdf*. This Interface Control Document (ICD) was created by Scalable Network Technologies and is distributed with their permission. Any communications effects server that conforms to this interface can be used with VR-Forces.

9.2. Configuring VR-Forces

By default, VR-Forces is configured to use a simple internal communications effects model that allows all messages to be passed to their recipients. To use an external communications effects server, you must:

- ♦ Enable the publishing of Radio Transmitters for any entity types that will be sending messages through the communications effects server
- ♦ Set the External Communications Model as the communications model to be used by VR-Forces
- ♦ Configure the External Communications Model.

9.2.1. Enabling Radio Transmitter Publishing

The External Communications Model requires that any entities sending or receiving messages through the communications effects server publish a DIS or HLA radio transmitter object. By default, VR-Forces entities do not publish this object.

To enable publishing of the radio transmitter object:

1. In the OPE file for each entity type for which you want to enable radio transmitter publishing, locate the **radios** block.
2. Change the **publish-transmitter** parameter to **True** for each radio, for example:

```
(radios
  (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)
    )
  )
)
```

To verify that transmitters are being published, create a scenario that has the entity types for which you enabled radio transmitter publishing. A radio icon should be displayed next to the entity symbol for each radio transmitter on the entity.

9.2.2. Enabling the External Communications Model

To enable the External Communications Model, edit the file *commModelParams.mtl*, which is in the top directory of the simulation model set that you are using. This file defines the communication models available to the scenarios. By default, VR-Forces is configured to use the **simple-radio-comm-model**, which allows all communication between entities to be received.

- To use an external server for all radio communication, comment out the **simple-radio-comm-model** block, and uncomment the **external-comm-model** block.

This causes the external server to be used for all radio communication.

9.2.3. Configuring the External Communications Model

If you enable the External Communications Model, you can set the following configuration options (in *commModelParams.mtl*):

- ♦ **use-actual-message-size** specifies how the size of the messages will be modeled. When a message is sent using the external communications model, a message size is sent to the external communications effects server, and that number may affect how long it takes for the message to reach its recipient. If this parameter is set to True (the default), the actual number of bytes of the VR-Forces message is used for this size value. This includes the VR-Forces message header of the sender or recipient and the contents of the message. If this parameter is set to False, the size specified by **substitute-message-size** is used for all messages.
- ♦ **substitute-message-size** specifies the message size, in bytes, to send to the external communications effects server. It is used only if **use-actual-message-size** is False.
- ♦ **message-timeout** specifies the timeout value, in seconds, for messages sent to the communications effects server. Messages that are not delivered to the server by the end of the timeout period are dropped.

9.3. Using QualNet

This section provides specific details about how to use QualNet as an external communications effects server with VR-Forces. VR-Forces must be configured to use the external communications effects server, as described in “[Configuring VR-Forces](#),” on page 9-2.

- ♦ QualNet only supports HLA 1.3 RPR FOM 1.0, so you must run the HLA 1.3 version of VR-Forces.
- ♦ You must install and configure the QualNet Standard Interfaces library for QualNet to interoperate with VR-Forces over HLA. Follow QualNet instructions for installing this module, and configuring it for use with the same RTI that is being used for VR-Forces.
- ♦ You must create a QualNet scenario that matches the VR-Forces scenario to be run using QualNet as the communications effects server. You can do this by hand, or using the QualNet Synchronizer program. Please refer to the QualNet documentation for such information.

9.3.1. Running VR-Forces with QualNet

To use VR-Forces with QualNet, the two programs must be loaded and scenarios started in the following sequence.

1. Start QualNet and load the desired QualNet scenario. (Right click the *.config* file, and choose Run.)
2. Start VR-Forces, and load the VR-Forces scenario.
3. After all entities have been displayed in the VR-Forces GUI, click the Play button on the QualNet interface.
4. Run the VR-Forces scenario.



- ♦ If you want to load a new VR-Forces scenario, you must stop the QualNet scenario, and load a new QualNet scenario using the procedure described in this section.
 - ♦ The VR-Forces rewind functionality does not work correctly with QualNet. To replay a scenario you must restart QualNet and VR-Forces as described in the procedure.
-

QualNet provides a program called *rprsynch* that can produce a QualNet scenario that corresponds to a VR-Forces scenario. See the QualNet documentation for details.

9.4. Example Scenario

VR-Forces includes an example VR-Forces scenario and a matching QualNet scenario to demonstrate the interoperability of VR-Forces and QualNet. The QualNet scenario was generated using *rprsynch*.

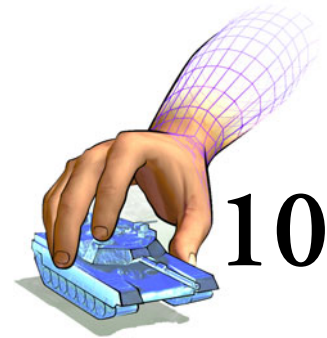
In the scenario, the M1A2 1 tank commands the UAV to fly over the area where the hostiles are and send back spot reports. When the first tank receives spot reports, it commands the second one to move to the waypoint, and also moves there itself. All commands and spot reports are sent through the radio, which is passed through QualNet. View the plans for M1A2 1 and UAV 1 to see the details of how they are communicating.

If none of the entities start moving after 10 seconds of running, there is probably an error communicating with QualNet.

9.4.1. Running the Scenario

To run the scenario, use the following steps:

1. Enable external communications modeling, as described in [“Configuring VR-Forces,”](#) on page 9-2.
2. Enable transmitter publishing for M1A2 1 and the generic-unmanned-aerial-vehicle, as described in [“Enabling Radio Transmitter Publishing,”](#) on page 9-3.
3. Copy the `./data/scenarios/QualNet/vrforces-CommDemo` folder to the `QualNet/scenarios` folder.
4. Run QualNet and run the `vrforces-CommDemo` scenario.
5. Start VR-Forces.
6. Open the `QualNet/commDemo` scenario.
7. Click the Play button on the QualNet interface.
8. Run the scenario in VR-Forces.



Configuring 3D Front End Communication

This chapter describes options that affect how and what the VR-Forces 3D Front End communicates over the network about the eyepoint. These options include network address settings and options for making the eyepoint visible on the network.

Communicating Information About the Eyepoint	10-2
Making the Eyepoint Visible to Other Applications.....	10-2
Configuring the Eyepoint Entity	10-3
Setting the DIS Protocol Version for Outgoing PDUs.....	10-5
Viewing and Attaching to Other Eyepoints	10-5
Controlling the Eyepoint from Other Applications	10-6
Enabling the Processing of View Control Messages.....	10-6
The Stealth Control Toolkit	10-6
Using Other MÄK Products to Control the Eyepoint.....	10-6
Using Mirroring.....	10-7
Sending View Control Messages Explicitly	10-7
Mirroring Display Options.....	10-8
Configuring the Frequency of View Control Messages	10-10
Configuring Multi-Channel Displays	10-11

10.1. Communicating Information About the Eyepoint

The VR-Forces 3D Front End can transmit state updates describing the eyepoint's position and orientation, and can respond to view control messages sent by other MÄK applications.

10.1.1. Making the Eyepoint Visible to Other Applications

By default, the 3D Front End sends state update messages to make its state information known to other applications. This makes it possible for a plan view display to represent the 3D Front End as an icon, or for one 3D Front End to synchronize its eyepoint movement with that of another 3D Front End.

To enable or disable broadcasting of eyepoint information:

1. Choose Options → Preferences. The Preferences dialog box opens.

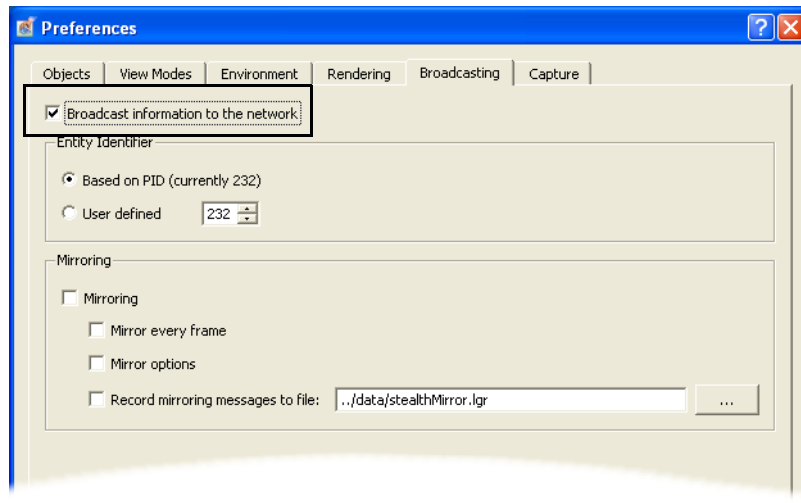


Figure 10-1. Preferences dialog box, Broadcasting tab

2. Select the Broadcasting tab.
 3. Check or clear Broadcast Information To The Network (Figure 10-1).
- To configure eyepoint broadcasting in *params.mtl*, edit the `enableStealthBroadcast` parameter.

10.1.2. Configuring the Eyepoint Entity

VR-Forces publishes (or sends) entity information to tell other applications about the eyepoint's current position and orientation. You can configure the following information for this entity:

- ♦ Marking text (DIS only)
- ♦ HLA name (HLA only)
- ♦ Force ID
- ♦ Entity ID
- ♦ Entity type
- ♦ Update thresholds.

Specifying the Marking Text

If you are sending update messages to the network, you may want to specify marking text to describe VR-Forces.

- To specify marking text, edit the **stealthMarking** parameter in *params.mtl*.

Specifying the HLA Name

The HLA name is the name of the broadcast entity for HLA. The default is MAK-Stealth.

- To specify the HLA name, edit the **stealthHLAName** parameter in *params.mtl*.

Setting the Force ID

The **stealthForceIdentifier** parameter determines the force ID that appears in entity state PDUs (DIS) and entity updates (HLA) that VR-Forces sends to the network.

The default force ID is 0.

- To specify the default force ID, edit the **stealthForceIdentifier** parameter in *params.mtl*.

Specifying the Entity Identifier

The entity Identifier (entity ID) identifies VR-Forces when it is sending update messages to the network. In DIS exercises, the entity ID is combined with the site ID and host ID to form the site:host:entity triplet. You can also use the entity ID to address view control messages to a particular 3D Front End. The default entity ID (set using the `stealthEntityIdentifier` parameter) is -1. If you use the default setting, 3D Front End uses the process ID to generate a random number that is used as the entity ID.

If you have enabled mirroring, each 3D Front End must have a unique entity ID. If a receiving 3D Front End has the same entity ID as the 3D Front End sending view controls, it will think they come from itself and will ignore them. The default entity ID setting usually ensures a unique entity ID. However, you if you want to, you can configure the entity ID yourself.

To specify the entity ID:

1. Choose Options → Preferences. The Preferences dialog box opens.
2. Select the Broadcasting tab.
3. Choose an option in the Entity Identifier group box (Figure 10-2). If you choose Based on PID, the 3D Front End generates a random number based on the PID. Using this method, it is likely that all 3D Front Ends will have unique entity IDs without further manual configuration. If you choose User-defined, you are responsible for ensuring that each 3D Front End in an exercise has a different entity ID.

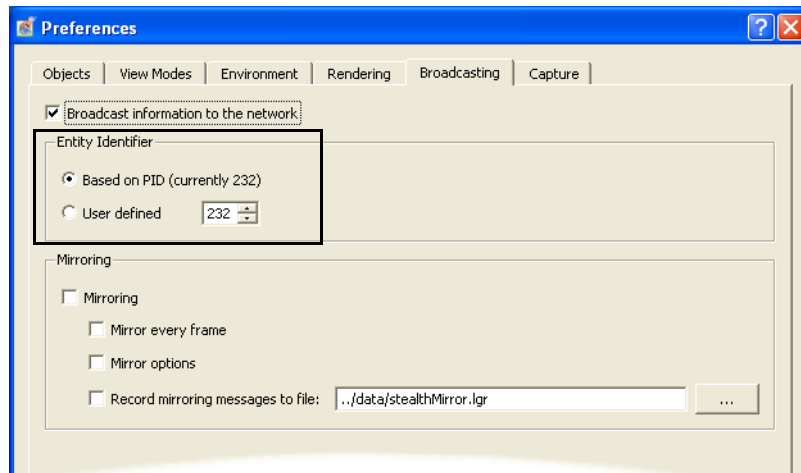


Figure 10-2. Preferences dialog box, setting VR-Forces ID

- To specify the entity ID, in *params.mtl* edit the `stealthEntityIdentifier` parameter.

Specifying the Entity Type

If you are sending eyepoint update messages to the network, you may want to specify a DIS or HLA RPR FOM entity type enumeration for the 3D Front End.

- To specify the entity type enumeration, edit the **stealthEntityType** parameter in *params.mtl*.

Specifying Update Thresholds

The update thresholds determine when the 3D Front End sends changes in orientation and rotation. The 3D Front End sends updates when changes in these parameters exceed the threshold.

- To specify update thresholds, edit the **stealthTranslationThreshold** and **stealthRotationThreshold** parameters in *params.mtl*.

10.1.3. Setting the DIS Protocol Version for Outgoing PDUs

- To set the DIS protocol version the 3D Front End specifies in outgoing PDUs, use the **protocolVersion** parameter.
- To override **protocolVersion**, use the **--disVersion** command-line option:

```
vrf3DGui -s 1 -a 3000 --disVersion n
```

where *n* is one of the following:

- 4 = DIS 2.0.4
- 5 = IEEE 1278.1
- 6 = DIS 2.1.4



While this setting affects the version number contained in the PDU, it does not affect the format of the PDU.

10.1.4. Viewing and Attaching to Other Eyepoints

If an exercise includes remote MÄK Stealths, VR-Forces 3D Front Ends, or both and they are sending state updates, VR-Forces can detect them. A remote eyepoint entity is listed in the object hierarchy, so you can attach the local eyepoint to it just as you would attach to any other entity.

Eyepoint icons are displayed in the VR-Forces 2D map, but they are not normally visible in the 3D Front End graphics display, even when they appear in the entity list. If you want to display visual representations of remote eyepoints, map a visual model to the eyepoint's entity enumeration.

10.2. Controlling the Eyepoint from Other Applications

VR-Forces can respond to view control messages generated by applications such as the MÄK Logger, VR-Forces 2D front-end, MÄK Stealth, and other 3D Front Ends (the mirroring feature). You can also write VR-Link-based applications to send view control messages to the 3D Front End using the classes in the Stealth Control Toolkit. You can use view control messages to:

- ♦ Select the view mode and attached entity
- ♦ Set the eyepoint's position and orientation
- ♦ Control the display of options such as trailing effects, track histories, and labels.



The view control PDUs are experimental, and are not an official part of DIS. They are of PDU kind 202.

10.2.1. Enabling the Processing of View Control Messages

- To enable or disable the processing of view control messages, choose Options → View Control.
- To enable or disable processing of view control messages at startup, edit the **enable-ViewControl** parameter in *params.mtl*.
- To specify that the 3D Front End should only respond to view control messages that are addressed to it, edit the **respondToMyViewControlOnly** parameter in *params.mtl*. View control messages contain a view ID that specifies the target application for the message. If **respondToMyViewControlOnly** is 1, the receiving application's entity ID is tested against the target viewer ID in the message.

10.2.2. The Stealth Control Toolkit

VR-Forces includes a set of libraries and header files that make up the Stealth Control Toolkit, which you can use to control 3D aspects of the 3D Front End. For details, please contact MÄK technical support.

10.2.3. Using Other MÄK Products to Control the Eyepoint

The MÄK Plan View Display and VR-Forces 2D front-end have options for attaching the eyepoint to entities.

10.3. Using Mirroring

Controlling several 3D views (slaves) from one 3D Front End (the master) is called mirroring. When mirroring is enabled, the 3D Front End sends view control messages whenever you change the view, the attachment state of an entity, and certain display options. You can also configure mirroring to respond to keyboard controls.

- To enable or disable mirroring in the GUI, choose Options → Mirroring.
- To enable mirroring at startup, use the `--mirror` command-line option.

10.3.1. Sending View Control Messages Explicitly

By default, the 3D Front End sends mirroring messages when you change the view mode or attachment using GUI controls. You can also send mirroring messages at any arbitrary point.

- To send a mirroring message, on the View Mode Information toolbar, click the Mirror button.

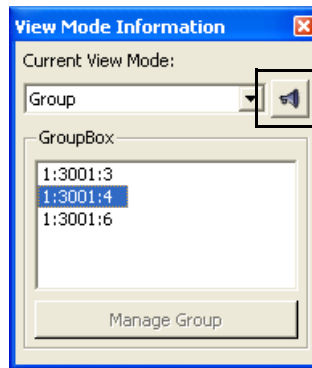


Figure 10-3. Mirror button

- To enable mirroring of views in *params.mtl*, set the `enableViewModeMirroring` parameter to 1.

10.3.2. Mirroring Display Options

The 3D Front End can send mirroring messages when you toggle options on the Options menu. For a list of display options that you can mirror, please see [“Mirroring Messages Supported,”](#) on page 10-9.

To enable or disable mirroring of display options:

1. Choose Options → Preferences. The Preferences dialog box opens.
2. Select the Broadcasting tab.
3. In the Mirroring group box ([Figure 10-4](#)), select the Mirror Options check box.

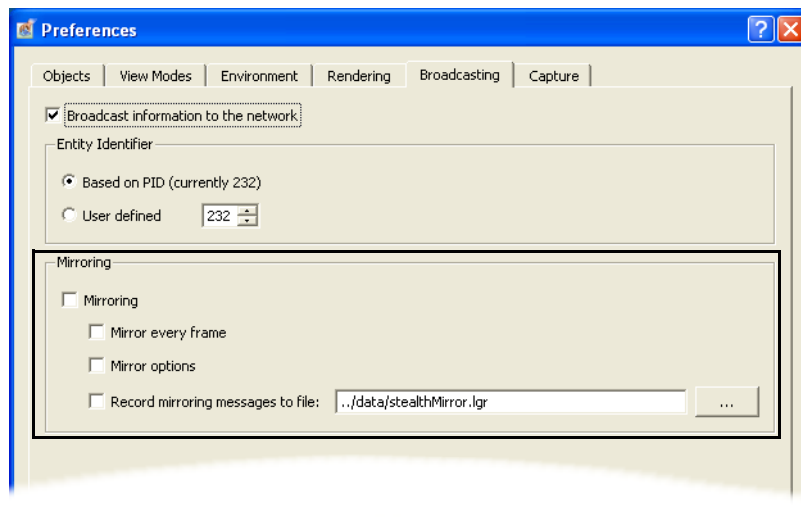


Figure 10-4. Preferences dialog box, mirroring options

Mirroring Messages Supported

The 3D Front End supports mirroring messages for the following options:

- ♦ Sound toggle
- ♦ Channel options
 - Texture toggle
 - Clouds toggle
 - Overlay toggle
 - Fog toggle
 - Sky dome toggle
 - Time of day
 - Date
- ♦ DIS/HLA
 - Smooth period
 - Smoothing toggle
- ♦ Label toggles
 - Entity
 - Clustering
 - Tactical graphics
- ♦ Trajectory histories
- ♦ Tactical graphics toggle
- ♦ Ground clamping toggle
- ♦ Ground clamp range
- ♦ Trailing effects toggle
- ♦ Load terrain - only absolute paths get sent
- ♦ XR mode
 - XR/Normal toggle
 - Height above terrain labels (entity only)
 - Current model set
 - Object scale
 - Terrain scale
 - Texture (MTD files only)
- ♦ Exiting the 3D Front End.

10.3.3. Configuring the Frequency of View Control Messages

By default, the 3D Front End only sends view control messages when you change the view mode or entity attachment in the GUI. If you want to send view control messages when you change the view from the keyboard, you need to enable sending of view control messages every frame.

To send view control messages every frame:

1. Choose Options → Preferences. The Preferences dialog box opens.
 2. Select the Broadcasting tab.
 3. In the Mirroring group box, select the Mirroring check box ([Figure 10-4](#)).
 4. Check Mirror Every Frame.
 5. Optionally, specify the file to use for saving the messages.
- To configure sending view control messages every frame in *params.mtl*, set the **send-MirrorMessagesEveryFrame** parameter to 1.

10.3.4. Configuring Multi-Channel Displays

The mirroring feature lets you set up a multi-channel configuration using multiple computers to display exercises. Typical configurations are a horizontal alignment, which widens the view, and the angled alignment, which surrounds the eyepoint (Figure 10-5). Since the 3D Front End just sends out its view, if you do not configure the displays for multi-channel, the slave monitors display the exact same view as the master. Therefore, for multiple views, you must configure the slave displays to calculate an offset from the master view. You must also ensure that each 3D Front End has a different ID, because the 3D Front End is configured by default to ignore messages that comes from itself, that is, messages that have the same ID as it does.

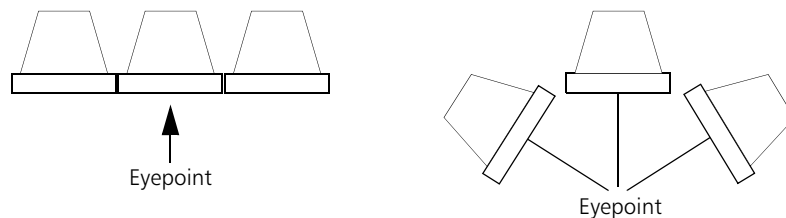


Figure 10-5. Multi-channel display

To configure a 3D Front End display for multi-channel, you must edit the *default.acf* configuration file.

To edit the *default.acf* file for multi-channel display:

1. Start the Lynx Prime application. (On Windows, choose Start → Programs → MAK Technologies → VR-Forces x.x → Tools → LynX Prime.)
2. Choose File → Open.
3. Select *default.acf* in the *./config* directory.
4. Select myChannel. The configuration settings are displayed (Figure 10-6).

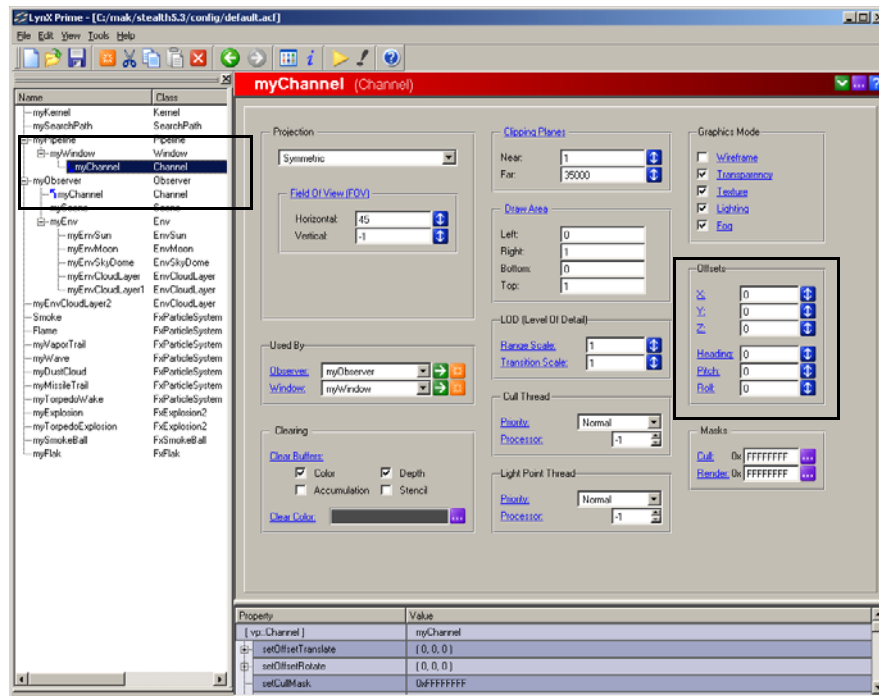
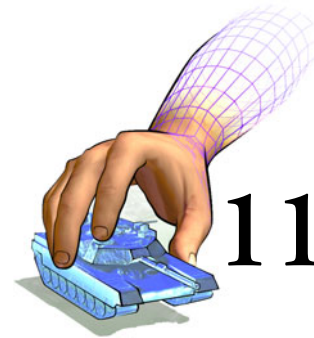


Figure 10-6. *default.acf* opened in LynX Prime

5. In the Offsets group box, set either the X, Y, and Z values or the Heading, Pitch, and Roll values. For example if you want to set up three monitors, with two side monitors at a 45° angle to the center monitor, set the headings for the side monitors to 45° and 315°. If you want to align monitors horizontally, set the side monitors to use positive and negative offsets of the X value.



3D Front End Configuration Files

You can customize the 3D Front End by modifying configuration files. This chapter describes these configuration files. It also discusses environment variables.

Specifying Parameters with the Startup Command.....	11-3
Setting Parameters from the Command Line	11-3
Setting Parameters by Loading a Configuration File	11-3
The Stealth.mtl File	11-4
The params.mtl File	11-5
Other Configuration Files	11-15
The ModelMap.mtl File	11-15
The Saved Views File	11-15
The aggregateMap.mtl File	11-15
Terrain Database Configuration Files	11-16
The trailMap.mtl File	11-16
The stealthSoundMap.mtl File	11-16
The tacticalGraphicsMap.mtl File	11-16
The clusterRulesMap.mtl File (XR mode Only)	11-16
The infantryMap.mtl File	11-16
The rangeRingsMap.mtl File	11-17
The hazardCloudMap.mtl File	11-17
The guipreferences.mtl File	11-17
Configuring Joysticks	11-17
Terrain Database Configuration Files	11-18
Specifying a Database's Origin	11-18
Multi-Part Databases	11-19
Specifying a Map Datum	11-19
Specifying the Default Eyepoint Position	11-20

Configuring TrueType Font-based Unit Icons (Windows Only) 11-20

 Unit Icon Parameters..... 11-21

 Symbol Mapping File for TTF-based Unit Icons..... 11-22

Editing the ACF Configuration File..... 11-23

11.1. Specifying Parameters with the Startup Command

The `-l` command-line option lets you:

- ♦ Set a parameter with the startup command, thereby overriding any corresponding value that was set by a configuration file
- ♦ Load an MTL configuration file.

The example commands in the following sections start the 3D Front End and set the `farFogRange` parameter to 150000 meters. The first example sets the parameter value directly; the second shows how to use an MTL command to set a parameter value.

11.1.1. Setting Parameters from the Command Line

- To set a parameter from the command line, use the `--lispFile` option followed by the parameter, an equals sign, and the argument, for example,

```
vrf3DGui -s 1 -a 3000 --lispFile farFogRange=150000.0
```

11.1.2. Setting Parameters by Loading a Configuration File

- To load additional (or alternative) configuration files at startup, use the `--lispFile` option as shown:

```
vrf3DGui -s 1 -a 3000 --lispFile myParams.mtl
```

The 3D Front End processes the specified file after loading *Stealth.mtl*, so the settings in your specified file will override corresponding settings in *Stealth.mtl*.

If you want to use the specified file instead of *Stealth.mtl*, use `--noDefaultMtl` as the first option on the command line.

11.2. The *Stealth.mtl* File

The *Stealth.mtl* file is loaded at startup unless you use the `-N` command-line option to prevent it from loading.

The primary job of the *Stealth.mtl* file provided by MÄK is to specify the pathname of other configuration files, and then to load those files. The default *Stealth.mtl* is as follows:

```
;; This is the main Configuration file that gets called when running
;; the Stealth. It uses MAK Technologies Lisp (mtl).
;; Set where other configuration files may be found

(setq mtl-path (list "../config"))
(setq font-path (list "../data/fonts"))
(load mtl-path "params.mtl")      ;; General parameter
(load mtl-path "ModelMap.mtl")    ;; Entity mappings
(load mtl-path "soundMap.mtl")    ;; Entity mappings
(load mtl-path "featureMap.mtl")  ;; FID code to model file mappings
(load mtl-path "trailMap.mtl")    ;; Trail mappings
(load mtl-path "hazardCloudMap.mtl") ;; hazard cloud mappings
(load mtl-path "rangeRingsMap.mtl") ;; range volumes map
(load mtl-path "tacticalGraphicsMap.mtl") ;; tactical graphics mappings
(load mtl-path "stealthSoundMap.mtl") ;; FOR VRF 3D front end only
(load mtl-path "vrf3DFrontEnd.mtl") ;; FOR VRF 3D front end only

;; Load this file if NOT using TTF unit icons (useTtfIcons in
params.mtl)

;;(load mtl-path "aggregateMap.mtl") ;; Aggregate symbols
;; This file is always loaded; when using TTF unit icons, it will only
;; be used for tactical graphics icons
(load mtl-path "symbolmap-bmp.mtl") ;; Symbols for the object hierarchy

;; Use this file to set which articulated part is attached to in turret
mode
(load mtl-path "artPartAttachMap.mtl")

;; Include this when using TTF Unit icons. Additional loads may be
;; added for user-defined fonts
(load-font font-path "MAK-MilStd-2525-b.ttf" )

;; XR Only Mappers
(if (equal XR 1)
    (sequence
      ;; Load this file when using TTF icons
      (load mtl-path "modelset-2525B-ttf.mtl")    ;; Alternate model set
      ;; Load this file if NOT using TTF icons
      ;;(load mtl-path "modelset-2525B.mtl")    ;; Alternate model set
      (load mtl-path "modelset-perspective.mtl") ;; Alternate model set
      (load mtl-path "clusterRulesMap.mtl") ;; cluster rules for grouping
    )
  )
)
```

11.3. The *params.mtl* File

The *params.mtl* file is the primary configuration file for the 3D Front End and determines many of the 3D Front End's default values. It also lets you control 3D Front End attributes that are not accessible through the GUI. [Table 11-1](#) lists the parameters in *params.mtl* in alphabetical order and briefly describes each one. It also lists some parameters that are usually in terrain database files. For more information, please see the comments in *params.mtl*. Parameters for joysticks and similar controllers, which are in *params.mtl*, are described in “[Configuring Joysticks](#),” on page 11-17. Some parameters let you specify different values for XR mode. The XR mode version of the parameter is listed with the regular version. Some XR mode parameters are described in Chapter 7, [The VR-Forces 3D Front End XR Mode](#), in *VR-Forces 3D Front End User's Guide*.

Table 11-1: 3D Front End configuration parameters (Sheet 1 of 10)

Parameter	Description
aggregateDisplayLevel xrAggregateDisplayLevel	For nested aggregates, specifies the number of levels to display.
aggregateModelRange	Specifies the radius around the eyepoint, in meters, in which aggregate models will be loaded. Default: 10000.0
aggregateTimeout	Specifies how long to display an aggregate entity after the most recent state update message was received.
appNumber	(DIS) Specifies the host portion of the entity ID.
blockAtNearClip	If set to 1, use the near clipping plane as the blocked distance. Default: 1.
blockedDistance	Specifies the distance, in meters, to maintain from blocking geometry when constrained by all geometry (viewConstraint is set to 2.) Default: 1.
date xrDate	Specifies the date. This is used for positioning the sun and moon.
defaultTimeThreshold	Specifies the DIS heartbeat. Default: 5.0 seconds.
destAddrString	Specifies the IP address to use for unicast.
destroyFedExec	If set to true (1), the <i>DtExerciseConn</i> destructor tries to destroy the federation execution. Default: 1.
detResultAirBurstColor	Specifies the color of the detonation of a munition that bursts in the air. Default: Black.
detResultDefaultColor	Specifies the color for all detonations that are not classified as entity impact, ground impact, or airburst. Default: White.
detResultEntityImpactColor	Specifies the color of a detonation that hits an entity. Default: White.
detResultGroundImpactColor	Specifies the color of the detonation of a munition that strikes the ground. Default: Gray.

Table 11-1: 3D Front End configuration parameters (Sheet 2 of 10)

Parameter	Description
<code>disPort</code>	Specifies the UDP port through which the 3D Front End sends and receives data.
<code>disVersion</code>	Specifies the DIS protocol version specified in outgoing PDUs.
<code>drainInputMaxTime</code>	Specifies the maximum time for reading DIS or HLA data, in seconds. A value of -1 indicates that all available messages should be processed. Default: 0.1.
<code>drainInputMaxTime</code>	Specifies a maximum time for reading DIS or HLA data. A value of -1 indicates that all available messages should be processed. Default: 0.1.
<code>emissionsColorIndicates</code> <code>xrEmissionsColorIndicates</code>	Specifies what the sensor volume color indicates.
<code>emissionsFarColor</code>	Specifies the color at the sensor volume's radius if <code>emissionsColorIndicates</code> is set to 2 (nothing). The color is specified with a hexadecimal number.
<code>emissionsHighFrequency</code>	Specifies the default high frequency, in Hz.
<code>emissionsHighPrFrequency</code>	Specifies the default high pulse repetition frequency, in Hz.
<code>emissionsLowFrequency</code>	Specifies the default low frequency, in Hz.
<code>emissionsLowPrFrequency</code>	Specifies the default low pulse repetition frequency, in Hz.
<code>emissionsNearColor</code>	Specifies the color at the sensor volume's origin if <code>emissionsColorIndicates</code> is set to 2 (nothing). The color is specified with a hexadecimal number.
<code>emissionsRadiusExponent</code> <code>xrEmissionsRadiusExponent</code>	Specifies the radius exponent, which applies exponential changes to the sensor volume radii.
<code>emissionsRadiusScaleFactor</code> <code>xrEmissionsRadiusScaleFactor</code>	Specifies the default radius scale, a number by which all sensor volume radii are multiplied.
<code>emissionsSegmentAngle</code> <code>xrEmissionsSegmentAngle</code>	Specifies, in degrees, the size of the arc represented by one segment.
<code>emissionsShowLines</code> <code>xrEmissionsShowLines</code>	Specifies whether or not segment outlines are displayed.
<code>emissionTimeout</code>	Specifies how long to display an emission after the most recent state update was received.
<code>enableAggregateRoutesDisplay</code>	Enables (1) or disables (0) the display of aggregate routes. Default: 1.
<code>enableAreasDisplay</code>	Enables (1) or disables (0) the display of areas. Default: 1.
<code>enableClouds</code> <code>xrEnableCloudLayer</code>	Enables (1) or disables (0) display of the cloud layer.

Table 11-1: 3D Front End configuration parameters (Sheet 3 of 10)

Parameter	Description
enableConsole	Enables (1) or disables (0) a display of messages to a console window.
enableDelayedModelLoading	Enables (1) or disables (0) delayed model loading.
enableDetonationLines	Enables (1) or disables (0) display of detonation lines.
enableEmissionsDisplay	Enables (1) or disables (0) display of emissions.
enableEntityLabels	Enables (1) or disables (0) the display of entity labels when you pass the mouse cursor over an entity.
enableFog	Enables (1) or disables (0) display of fog.
xrEnableFog	
enableFullScreen	Enables (1) or disables (0) starting the 3D Front End in full screen mode.
enableGroundClamping	Enables (1) or disables (0) ground clamping.
xrEnableGroundClamping	
enableGroundOrientationClamping	Enables (1) or disables (0) orientation clamping.
xrEnableGroundOrientationClamping	
enableLifeformsAlwaysUpright	When enabled, the orientation of lifeforms is adjusted to ensure that they are always upright, regardless of the terrain.
enableLighting	Enables (1) or disables (0) lighting effects.
xrEnableLighting	
enableMouseLook	Enables (1) or disables (0) use of the mouse wheel to move the eyepoint forward and backward. Default: 1.
enableObstaclesDisplay	Enables (1) or disables (0) the display of obstacles. Default: 1.
enableOverlay	Enables (1) or disables (0) display of the overlay.
enableOverlayLogo	Enables (1) or disables (0) display of a bitmap logo in the overlay.
enableOverlayObjectsDisplay	Enables (1) or disables (0) the display of overlay objects. Default: 1.
enablePhaseLinesDisplay	Enables (1) or disables (0) the display of phase lines. Default: 1.
enablePivotGraphic	Enables (1) or disables (0) display of the Pivot mode graphic.
enablePlugins	Enables (1) or disables (0) loading of plug-ins at startup.
enableRemoteGraphics	Enables (1) or disables (0) drawing of remote graphics.
enableRoutesDisplay	Enables (1) or disables (0) the display of routes. Default: 1.

Table 11-1: 3D Front End configuration parameters (Sheet 4 of 10)

Parameter	Description
enableSkyDome xrEnableSkyDome	Enables (1) or disables (0) display of the sky dome.
enableSound	Enables (1) or disables (0) sound effects.
enableStealthBroadcast	Controls the visibility of the 3D Front End on the network.
enableTacticalGraphicLabels	Enables (1) or disables (0) the display of tactical graphics labels. Default: 1.
enableTacticalGraphicsDisplay	Enables (1) or disables (0) the display of tactical graphics. Default: 0.
enableTexture xrEnableTexture	Enables (1) or disables (0) texture effects.
enableTrailingEffects xrEnableTrailingEffects	Enables (1) or disables (0) display of trailing effects.
enableTrajectoryHistories xrEnableTrajectoryHistories	Enables (1) or disables (0) display of track histories.
enableTransparency xrEnableTransparency	Enables (1) or disables (0) transparency effects.
enableViewControl	Controls processing of view control messages.
enableViewModeMirroring	Enables (1) or disables (0) sending of view control messages when the view mode or attached entity changes.
enableViewModeRecording	Enables (1) or disables (0) recording of view mode changes to a MÄK Data Logger file.
enableViewModeStatusMes- sages	Enables (1) or disables (0) sending of status messages about its view mode.
enableWaypointsDisplay	Enables (1) or disables (0) the display of waypoints. Default: 1.
enableWireframe xrEnableWireframe	Enables (1) or disables (0) wireframe mode.
entityModelRange	Specifies the radius around the eyepoint, in meters, in which models will be loaded.
entityTimeout	Specifies the length of time, in seconds, during which the 3D Front End continues to display an entity after the last entity update message is received.
execName	Specifies the default federation execution name.
exerciseld	Specifies the exercise ID that the 3D Front End should listen to.
farClippingPlane xrFarClippingPlane	Specifies the distance, in meters, from the eyepoint beyond which entities and terrain objects are not processed. For XR mode, set this to a large number.

Table 11-1: 3D Front End configuration parameters (Sheet 5 of 10)

Parameter	Description
farFogRange	Specifies the initial value, in meters, of the Far Fog slider.
xrFarFogRange	
federateName	Specifies the federate name for HLA.
fedFileName	Specifies a FED file name. If not specified, a FED file matching the executive name is loaded. For example, if the executive name is VR-Link, the 3D Front End loads <i>VR-Link.fed</i> . Default: "".
filterFomClasses	Specifies a list of FOM classes that the 3D Front End should ignore.
filterFomInteractions	Specifies a list of interaction classes that the 3D Front End should ignore.
fireDetonateVisualsRange	Specifies the radius around the eyepoint, in meters, in which explosions and flak are displayed for detonate and fire interactions.
fomMapperInitData	Specifies a string used to initialize a FOM Mapper.
fomMapperLib	Specifies a FOM Mapper library.
forceFriendlyAggregateColor	Specifies the color used for aggregate volumes (enclosures) for friendly entities.
forceFriendlyDetonationColor	Specifies the color of the line that connects a friendly force entity to the detonation of a munition that it has fired. Default: Blue.
forceFriendlyHistoryColor	Specifies the color used for trajectory histories for friendly entities.
forceFriendlyLabelColor	Specifies the color for friendly force labels. Default: Blue.
forceNeutralAggregateColor	Specifies the color used for aggregate volumes (enclosures) for neutral entities.
forceNeutralDetonationColor	Specifies the color of the line that connects a neutral force entity to the detonation of a munition that it has fired. Default: Green.
forceNeutralHistoryColor	Specifies the color used for trajectory histories for neutral entities.
forceNeutralLabelColor	Specifies the color for neutral force labels. Default: Green.
forceOpposingAggregateColor	Specifies the color used for aggregate volumes (enclosures) for opposing entities.
forceOpposingDetonationColor	Specifies the color of the line that connects an opposing force entity to the detonation of a munition that it has fired. Default: Red.
forceOpposingHistoryColor	Specifies the color used for trajectory histories for opposing entities.
forceOpposingLabelColor	Specifies the color for opposing force labels. Default: Red.

Table 11-1: 3D Front End configuration parameters (Sheet 6 of 10)

Parameter	Description
<code>geodeticMapDatum</code>	Specifies a map datum other than WGS84. Usually placed in the terrain database configuration file.
<code>gradientFadeLabelColor</code>	Specifies the color that the entity label fades to. Default: semi-transparent black.
<code>groundClampDetonationTerrain-Hits</code>	Enables (1) or disables (0) ground clamping of detonations on the terrain. Default: 1.
<code>groundClampingMinimumExtent</code> <code>groundClampingMaximumExtent</code>	Specifies a range, in meters, in which the 3D Front End checks for surfaces to clamp the entity to, starting from the topmost to the bottommost.
<code>groundClampThreshold</code> <code>xrGroundClampThreshold</code>	Specifies the default range of interest for ground clamping. Enter the square of the desired range, in meters. Default: 50 meters.
<code>intervisibilityLineColor</code>	Specifies the color used for intervisibility lines. Default: 0xFF00FF00 (green).
<code>intervisibilityRange</code> <code>xrIntervisibilityRange</code>	The distance, in meters, within which intervisibility is calculated.
<code>isectorUpdateThreshold</code>	Specifies the distance, in meters, an entity needs to travel before recalculating its ground clamped elevation and orientation. Default: 0.1.
<code>labelFontSize</code>	Specifies the font size for box labels. Default: 14.
<code>labelStyle</code>	Specifies the label style. 0 = text 1 = box. Default: 1.
<code>loadAsynchronous</code>	Load terrains synchronously (0) or asynchronously (1). Synchronous loading is faster than asynchronous.
<code>lodScale</code> <code>xrLodScale</code>	Configures the switch range for levels of detail in the display of entity models.
<code>maxNumberOfAggregateModels</code>	Specifies the maximum number of aggregate models that can be displayed. If more than N models fit within the desired range, the Nth closest are displayed. Default: 100.
<code>maxNumberOfEntityLabels</code>	Specifies the maximum number of entity labels to draw. If this number is fewer than the total number of entities and delayed models loading is enabled, the closest loaded models are drawn first. Otherwise, labels are drawn in the order in which entities are discovered.
<code>maxNumberOfEntityModels</code>	Specifies the maximum number of loaded models. If more than N models fit within the desired range, the Nth closest are displayed.

Table 11-1: 3D Front End configuration parameters (Sheet 7 of 10)

Parameter	Description
maxNumberOfFireDetonateEffects	Specifies the maximum number of loaded effects that can be displayed. If more than N effects fit within the desired range, the Nth closest are displayed. Default: 100.
maxNumberOfInsetViews	Specifies how many inset views you can open at one time.
maxNumberOfTrailModels	Specifies the maximum number of trail segments that can be in the scene at one time for any particular trail type.
mcastTtl	Specifies the multicast time-to-live value for DIS. Default: 2.
nearClippingPlane xrNearClippingPlane	Specifies the distance, in meters, from the eyepoint within which entities and terrain objects are not processed.
notifyLevel	Specifies the notification level for messages printed to the console and saved in <i>vrfGui.log</i> . 0 - Fatal messages only 1 - Warning messages 2 - Informational messages 3 - Debug messages 4 - Verbose (all messages). Default: 2.
numTtfIconForces	Specifies the number of forces that will be supported when creating mappings for TrueType font generated icons. Default: 16.
overlayColor	Specifies the color used to display the overlay.
overlayCompanyName	Specifies the name displayed in the overlay in the company name field. Default: MÄK Technologies.
overlayLogoFile	Specifies the bitmap file to display as the overlay logo.
overlayLogoHeight	Specifies a scale factor for the height of the logo. Default: 0.3.
overlayLogoPreserveSize	If set to 1, preserve the size of the overlay logo, no matter what the size of the display. Default: 1.
overlayLogoWidth	Specifies a scale factor for the width of the logo. Default: 0.3.
overlayProductName	Specifies the name displayed in the overlay in the product name field. Default: VR-Forces.
pivotGraphicModel	Specifies the OpenFlight model to use for the pivot graphic.
preloadModels	Specifies loading of models at startup, or dynamic model loading (no loading at startup.)
recvBufferSize	Specifies the netsocket receive buffer size.

Table 11-1: 3D Front End configuration parameters (Sheet 8 of 10)

Parameter	Description
<code>respondToMyViewControlOnly</code>	Specifies whether the 3D Front End should respond to all view control messages or only those addressed to this specific 3D Front End.
<code>rprFomVersion</code>	Specifies the version of the RPR FOM being used.
<code>sendBufferSize</code>	Specifies the netsocket send buffer size.
<code>sendFedTime</code>	If set to true (1), interactions will be sent or received in timestamp order. Default: 0.
<code>sendMirrorMessagesEvery- Frame</code>	Enables (1) or disables (0) sending mirror messages every frame. When enabled, provides constant updates to slave VR-Forcess.
<code>sendMirrorMessagesForOptions</code>	Enables (1) or disables (0) sending option PDUs when mirroring is on.
<code>showHighestAggregatesOnly</code> <code>xrShowHighestAggregatesOnly</code>	Enables (1) or disables (0) showing only the highest level aggregate.
<code>siteld</code>	(DIS) Specifies the site portion of the entity ID.
<code>smoothPeriod</code> <code>xrSmoothPeriod</code>	Specifies the period of time, in seconds, over which trajectory smoothing takes place.
<code>soundRange</code>	Specifies the range, in meters, around the eyepoint in which sounds are played. Default: 500.0.
<code>soundRangeEnabled</code>	Enables (1) or disables (0) the <code>soundRange</code> parameter. Default: 1.
<code>startAltitude</code>	Specifies the starting altitude for the eyepoint. Usually placed in the terrain database configuration file.
<code>startOrientation</code>	Specifies the starting orientation for the eyepoint. Usually placed in the terrain database configuration file.
<code>startPosition</code>	Specifies the starting position for the eyepoint. Usually placed in the terrain database configuration file.
<code>stealthEntityIdentifier</code>	Specifies a unique identifier for the 3D Front End VR-Forces. Used in DIS to create the site:host:entity identifier. Used in HLA to create a unique identifying string and as an entity identifier.
<code>stealthEntityType</code>	Specifies the DIS/RPR FOM entity type enumeration for the 3D Front End.
<code>stealthForceIdentifier</code>	Specifies the Force ID in DIS entity state PDUs and HLA entity updates. Specify an integer.
<code>stealthHLAName</code>	Specifies the name of the 3D Front End broadcast entity for HLA. Default: MAK-Stealth.

Table 11-1: 3D Front End configuration parameters (Sheet 9 of 10)

Parameter	Description
stealthMarking	Specifies descriptive text about the 3D Front End that is sent out with update messages if the 3D Front End is visible on the network. In the 3D Front End, the marking text is displayed in the Markings field of the Entity Information dialog box.
stealthRotationThreshold	For DIS, sets the threshold for eyepoint rotation at which the 3D Front End sends updates.
stealthTranslationThreshold	For DIS, sets the threshold for eyepoint translation at which the 3D Front End sends updates.
tacticalGraphicTimeout	Specifies the length of time, in seconds, during which the 3D Front End continues to display tactical graphics after the last update message is received. Default: 100.0.
textLabelColor	Specifies the color for text labels. Default: 0xFFFFFFFF.
timeOfDay xrTimeOfDay	Specifies the time of day, which controls the quality of ambient light.
trailsBasedOnAppearance	Specifies whether or not to draw trailing effects on a per entity basis, based on the Trailing Effects attribute.
trajectoryHistoryLength	Specifies the maximum length of a trajectory history, in meters.
ttfSymbolMap	Specifies the symbol map file used for generating TrueType font icons in the GUI and the main window. Default: symbolMap.csv.
updateHiddenTrajectories	Allows you to update trajectory histories while they are hidden. When histories are not updated, the history is cleared when it is turned off.
useAbsoluteTimeStamps	Specifies use of absolute (1) or relative (0) timestamps on messages. Default: 0.
useAdvisories	If set to true (1), VR-Link sends updates for classes, attributes, and interactions based on advisories describing the existence of remote subscribing federates. Default: 1.
useAsyncIO	Enables (1) or disables (0) asynchronous I/O for DIS. Default: 1.
useTtfIcons	Specifies use of TTF icons (1), or FLT file or BMP files (0) for unit icons. Default: 1.
verticalFieldOfView xrVerticalFieldOfView	Specifies the vertical field of view in degrees. The horizontal field of view depends on the display's aspect ratio.

Table 11-1: 3D Front End configuration parameters (Sheet 10 of 10)

Parameter	Description
viewConstraint	Sets the default view constraint mode (for Absolute mode only), where: 0 = unconstrained 1 = constrained by view floor 2 = constrained by all geometry. Default: 0.
viewDefaultOffset	Specifies the distance, in meters, from the attached entity for a view mode's reset position.
viewDefaultPitchOffset	Specifies the pitch of the eyepoint, in degrees, when the eyepoint is reset.
viewFloor	Specifies the height, in meters, above the terrain in Absolute or Track mode, below which the eyepoint cannot go.
viewStepSize xrViewStepSize	Specifies the starting step size, in meters, when the eyepoint moves left, right, forward, back, up, or down.
viewTurnRate xrViewTurnRate	Specifies the starting turn rate, in degrees, when the eyepoint turns left, right, up, or down.
viewWideGroupMaxPsiRate	Specifies the maximum rate, in radians/sec, that the view's heading can change when it attempts to position itself for Wide Group mode. The heading is set based on an algorithm that attempts to get maximum separation (in the graphics display) of the entities which are farthest apart in the 3D world.
whoToSmooth xrWhoToSmooth	Specifies types of objects to which smoothing applies.

11.4. Other Configuration Files

This section describes configuration files other than *Stealth.mtl* and *params.mtl*.

11.4.1. The *ModelMap.mtl* File

The *ModelMap.mtl* file contains `model-map` commands for all the entity models supplied with the 3D Front End. Model mapping and the `model-map` command is explained in Chapter 12, [Configuring 3D Entity Models](#).

You can also specify whether or not to load models at startup, or load them dynamically. In *params.mtl*, use the `preloadModels` parameter, as follows:

```
(setq preloadModels mode)
```

where:

- ♦ 0 = Do not load models at startup
- ♦ 1 = Load models at startup.

The *ModelMap-billboards.mtl* File

The *ModelMap-billboards.mtl* file contains model mappings for the billboard models (2-D entity icons.)

11.4.2. The Saved Views File

When you save views, they are written to file as a series of commands similar to the following:

```
(viewsave "2-Compass-T80 1" 2 "1:3001:579,20.687752,  
-9.773844,0.174533,0.000000,0.000000,0.000000,T80 1")  
(viewsave "8-Orbit-Squad 1" 5 "1:3001:611,-48.309662,0.087266,  
-2.059527,0.087266,0.000000,0.000000,0.000000,Squad 1")
```

You would not normally build a saved view file by hand, nor would you edit one by hand. Saving views is discussed in Section 3.3, [“Saving and Recalling Views”](#), in *VR-Forces 3D Front End User’s Guide*.

11.4.3. The *aggregateMap.mtl* File

The *aggregateMap.mtl* file defines models for aggregates. This file is not compatible with the Model Mapper tool. If you want to edit it, you must edit it by hand. The aggregate mapping parameters are described in [“The aggregateMap.mtl File,”](#) on page 12-4.

11.4.4. Terrain Database Configuration Files

The 3D Front End includes configuration files for several terrain databases, including, *kabul.mtl*, *makland.mtl*, and *monterey.mtl*. For details about how to create a database configuration file, please see [“Terrain Database Configuration Files,”](#) on page 11-18.

11.4.5. The *trailMap.mtl* File

The *trailMap.mtl* file maps trailing effects models. For details, please see the comments in the file and [“The *trailMap.mtl* File,”](#) on page 12-4.

11.4.6. The *stealthSoundMap.mtl* File

The 3D Front End’s sound mapping commands let you associate sounds with entities such as vehicles and munitions, and with events such as collisions, munition firings, and detonations. To configure sound effects you add **entity-sound**, **fire-sound**, and **detonate-sound** parameters to the *stealthSoundMap.mtl* file.

The sample sound configuration file, *./config/stealthSoundMap.mtl*, contains a set of predefined sound mappings for common battlefield entities and events. You need to modify these mappings whenever you add or delete a sound. For instructions, please see the comments in *stealthSoundMap.mtl*.

The *params.mtl* file has an option (**soundRange**) for specifying the range from the eyepoint within which entity sounds are played. Adjusting this setting can improve performance.

11.4.7. The *tacticalGraphicsMap.mtl* File

The *tacticalGraphicsMap.mtl* file configures the supported tactical graphics. Please see the comments in the file for a description of the format.

11.4.8. The *clusterRulesMap.mtl* File (XR mode Only)

The *clusterRulesMap.mtl* file specifies rules for choosing the model to use when clustering entities in XR mode. You specify the paths for the models that you want to use. Then you create lists of entity enumerations that use that model.

11.4.9. The *infantryMap.mtl* File

The *infantryMap.mtl* file contains model mappings for dismounted infantry. It uses the basic model mapping syntax and has no special parameters.

11.4.10. The *rangeRingsMap.mtl* File

The *rangeRingsMap.mtl* file configures range volumes. You can add entity types for entities for which you want to display range volumes. You can specify the size of the volume, the color, transparency, and fill pattern. Please see the comments in the file for syntax details.

11.4.11. The *hazardCloudMap.mtl* File

The *hazardCloudMap.mtl* file configures hazard clouds. Please see the comments in the file for syntax details.

11.4.12. The *guipreferences.mtl* File

The *guipreferences.mtl* file saves the window layout and settings in the Preferences dialog box when you exit. It gets overwritten each time you exit.

11.5. Configuring Joysticks

The 3D Front End uses the same configuration for joysticks as the 2D front-end. You cannot manipulate the eyepoint with a joystick. You can only use a joystick to maneuver entities. Of course, if you attach the eyepoint to an entity, as you manage the entity, the eyepoint will follow along.

11.6. Terrain Database Configuration Files

The 3D Front End can load Presagis OpenFlight files (FLT files). To load a FLT file, you must provide information in a configuration file. The configuration file does the following:

- Specifies a list of database parts (a part is a group of files stored at one location), and associates this list of parts with a database name.
- For UTM, Flat, or LCC databases, sets the 3D Front End's origin to match the origin of the database.

For example, the file *hunter.mtl* contains the following commands that perform the actions to load the Hunter-Ligget database:

```
;; Specify database parts
(setq hunter-95-DB (list (list
  "/data/terrains/hunter95;../data/terrains/hunter95" "HL_94.flt")))
;; Specify the name for the database
(setq databaseName hunter-95-DB)
;; Specify the database type. Can be UTM, Lambert, FlatEarth, or
  Geocentric.
(setq databaseType "UTM")
;; Specify the origin of the database
(setq databaseOrigin "35:59:59.991:N,123:0:0.131:W,0")
```

This series of commands can be contained in any configuration file that the 3D Front End loads at startup. We recommend that you create a separate configuration file for each database you use, then put a single command in *Stealth.mtl* file to load the desired database configuration file.

11.6.1. Specifying a Database's Origin

This section describes the parameters and format to specify the database origin for UTM, geocentric, Lambert, and flat earth databases. The database type must be specified exactly as shown.

Geocentric:

Optionally, specify a local origin offset:

```
(setq databaseOrigin "0.0,0.0,0.0")
(setq databaseType "Geocentric")
```

UTM

Specify latitude, longitude, use of standard offsets (0=no,1=yes), and optionally, a zone:

```
(setq databaseOrigin "36:0:0:N,123:0:0:W,0,10")
(setq databaseType "UTM")
```

Lambert

Specify latitude, longitude, upper parallel, and lower parallel:

```
(setq databaseOrigin "36:0:0:N,123:0:0:W,37:0:0:N,35:0:0:N")
(setq databaseType "Lambert")
```

Flat Earth

Specify latitude and longitude:

```
(setq databaseOrigin "36:0:0:N,123:0:0:W")
(setq databaseType "FlatEarth")
```

11.6.2. Multi-Part Databases

A database part is a list containing a path and one or more MultiGen-Paradigm FLT files. If your database includes files in multiple directories, put multiple inner `list` commands into your database definition, as in this example of a three-part database:

```
(setq hunter-95-DB
  (list
    (list "./mydata/dir-2" "A-patch.flt")
    (list "./mydata/dir-2" "B-patch.flt" "C-patch.flt")
    (list "./mydata/dir-3" "D-patch.flt")
  )
)
```

11.6.3. Specifying a Map Datum

The `geodeticMapDatum` parameter defines the reference ellipsoid to be used in coordinate conversions. The default reference ellipsoid is WGS84. Map datums consist of the semi major and semi minor axes of the ellipsoid followed by an origin offset with respect to WGS84. Specify these values in meters. The sample terrain database configuration file, *hunter.mtl*, has several common reference ellipsoids defined that you can use as templates for your own custom configurations.

► To specify a different datum, use a command similar to:

```
(setq datumED50 "6378388.0 6356911.946127 -87. -98. -121.")
(setqb geodeticMapDatum datumED50)
```

The first command assigns a name (`datumED50`) to the datum. The second causes the 3D Front End to use that datum. These commands can be included in any configuration file that is loaded at startup, but you may find it most convenient to put them in the file that loads your database.

Please see the *hunter.mtl* configuration file for specifications of WGS84, ED50, NASANAD27, NASBNAD27 and CONUSNAD27 datums.

11.6.4. Specifying the Default Eyepoint Position

As part of configuring a database, you can specify a start position for the eyepoint. The default eyepoint position is set to an altitude of 500 meters, in the center of the database, facing north.

- To specify the eyepoint's altitude at startup, edit the **startAltitude** parameter.
- To specify the eyepoint's position at startup, edit the **startPosition** parameter.
- To specify the eyepoint's orientation at startup, edit the **startOrientation** parameter.

11.7. Configuring TrueType Font-based Unit Icons (Windows Only)

The 3D Front End supports generation of unit icons from TrueType fonts (TTF). The 3D Front End includes a font that implements MILSTD 2525B icons. Unit icon bitmaps are created by composing fill, outline, and if needed, echelon characters from the font. You can use other fonts too. The bitmaps are used in the GUI dialog boxes and panels that display unit icons. They are converted to textures and applied to billboards for use with aggregate bounding boxes and the 2525B model set in XR mode.

The 3D Front End uses a symbol mapping file to map DIS and RPR FOM entity types to the fill and outline characters used. The file provides mappings for a significant number of entity and aggregate types.

The 3D Front End also supports use of *.bmp* files for the GUI and *.flt* files for the 3D scene for creating icons. You can use these methods by editing *stealth.mtl* and *params.mtl*. However, the font-based unit icons comply more closely with MILSTD 2525B.

The characters in the MILSTD 2525B font are illustrated in Appendix A, [MILSTD 2525B Font Characters](#), in *VR-Forces 3D Front End User's Guide*.

11.7.1. Unit Icon Parameters

Parameters in *Stealth.mtl* and *params.mtl* determine how unit icons are created and specify where the supporting files are located.

The unit icon-related parameters in *params.mtl* are:

- ♦ **useTtfIcons** specifies the use of TTF unit icons (1), or *.flt* or *.bmp* unit icons (0).
- ♦ **numTtfIconForces** specifies which force types it should pre-render unit icons for. The TTF unit icon code can support up to 32 forces. However, since the 3D Front End only supports other, friendly, opposing, and neutral, there is no reason to change the default value (4).
- ♦ **ttfSymbolMap** specifies the symbol mapping file to use for mapping entity and aggregate types to the characters taken from the TrueType font.

Stealth.mtl manages the loading of files needed for unit icon generation, as follows:

- ♦ **font-path** specifies where the 3D Front End should look for the fonts that the `load-font` command specifies, for example:

```
(setq font-path (list "../data/fonts"))
```
- ♦ **load-font** tells the 3D Front End to load a TrueType font. In the following example, the MILSTD 2525B font provided by MAK is loaded.

```
(load-font font-path "MAK-MilStd-2525-b.ttf" )
```
- ♦ XR modesupports multiple model sets. If you are using TTF unit icons, load *modelset-2525B-ttf.mtl*. Otherwise, load *modelset-2525B.mtl*, as shown in the example:

```
(load mtl-path "modelset-2525B-ttf.mtl")  
(load mtl-path "modelset-2525B.mtl")
```
- ♦ If you are not using TTF unit icons, *aggregateMap.mtl* is used to map between aggregate types and the icons used above aggregate bounding boxes in the 3D scene. No additional mapping file is needed if you are using TTF unit icons.

```
(load mtl-path "aggregateMap.mtl")
```

11.7.2. Symbol Mapping File for TTF-based Unit Icons

The file used to map entity and aggregates types to characters is *symbolMap.csv*. It has the following fields:

- ♦ Entity Name
- ♦ Entity Kind
- ♦ Entity Domain
- ♦ Entity Country
- ♦ Entity Category
- ♦ Entity Subcategory
- ♦ Entity Specific
- ♦ Entity Extra
- ♦ NOT USED
- ♦ NOT USED
- ♦ Font Family (MAK MilStd 2525b is the font family name of the font provided)
- ♦ Outline Character
- ♦ Fill Character
- ♦ Modify Fill Character By Force flag.

The method for mapping an entry to a given entity or aggregate type is the same as for all our other mapping files.

- ♦ Outline and fill characters are specified as 2-byte hexadecimal values. An outline character is always specified.
- ♦ A fill character is optional - if not provided, one is chosen based on the entity's domain (mappings provided below), except for munitions, which have their own default mappings.
- ♦ If the Modify Fill Character By Force flag is true, then the fill character provided should be the correct fill character for the indicated entity if its force type is 0. The font is set up in such a way that adding the force type to the specified fill character provides the correct fill character for an entity from that force. If the Modify Fill Character by Force flag is false, then the same fill character (the one specified) is used for all forces.

Table 11-2 lists the default fill character mappings.

Table 11-2: Default fill character mappings

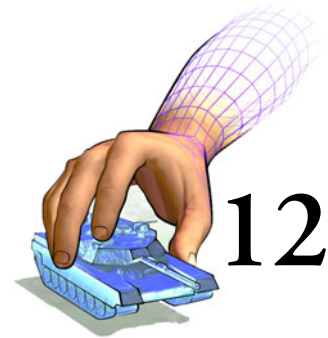
	forceOther	forceFriendly	forceNeutral	forceOpposing
munitions	0021	0022	0023	0024
air	0021	0022	0023	0024
space	0021	0022	0023	0024
land	0025	0026	0027	0028

Table 11-2: Default fill character mappings

	forceOther	forceFriendly	forceNeutral	forceOpposing
surface	0025	0029	0027	0028
subsurface	002A	002B	002C	002D

11.8. Editing the ACF Configuration File

In addition to MTL configuration files, the 3D Front End, because it is built using Vega Prime, has an ACF file. You can edit ACF files by hand or with MultiGen-Paradigm Lynx Prime, which is distributed with the 3D Front End. Some features that you can configure in the ACF file also have MTL parameters. In such cases, the MTL values override the ACF values. If you want the values in the ACF file to take effect, comment out the comparable parameters in the MTL file.



Configuring 3D Entity Models

This chapter discusses 3D visual models. Chapter 13, *Using the Model Mapper* explains how to create model map files with the Model Mapper utility.

How the 3D Front End Maps Model Files to Entities	12-2
The model-map Command	12-2
Defining Symbolic Names for Entity Types and Models	12-3
Model Mapping Files Supplied with VR-Forces	12-4
The ModelMap.mtl File	12-4
The aggregateMap.mtl File	12-4
The trailMap.mtl File	12-4
The modelset-2525B.mtl Files (XR Mode Only)	12-6
Adding Your Own Model Mappings	12-6
Custom Model Mapping Files	12-6
Articulated Parts and Entity Appearances	12-7
Models Supplied with the 3D Front End	12-8

12.1. How the 3D Front End Maps Model Files to Entities

Model mapping is the process of associating an entity type enumeration with a 3D model contained in a OpenFlight (FLT) data file. This is done through a series of `model-map` commands in one or more configuration files.

An entity type enumeration is a list of seven components as specified by the DIS protocol and HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific, 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.

Most model mapping is in *ModelMap.mtl*. Although this chapter explains in detail the model mapping process, we recommend that you use the Model Mapper to edit *ModelMap.mtl*. If you want to create model mapping files by hand, we recommend that you give them different names and load them from a configuration file.

12.1.1. The model-map Command

Models are mapped to objects with `model-map` commands. A `model-map` command is structured as follows:

```
(model-map (list entity-type) (list model))
```

where:

- *entity-type* is an entity type enumeration. It can represent a single entity type or a class of entity types (through the use of wildcards in the enumeration.)
- *model* refers to the name of the FLT file containing the image that will represent the entity.

In the configuration files supplied with VR-Forces, models are mapped using either a one-command format or a multi-command format that uses symbolic names. The multi-command format is somewhat easier to read. The use of symbolic names is explained in detail in [“Defining Symbolic Names for Entity Types and Models,”](#) on page 12-3.

A one-command `model-map` command could look like:

```
(model-map (list 1 2 -1 -1 -1 -1 -1)
  (list ".\data\Monterey\Moving_models" "smart_hog.flt"))
```

In this example, the entity-type enumeration defines a class of all platform-air vehicles. This broad class is defined by using the wildcard (-1) in several fields. These entities will be displayed using the specified graphics model file, which contains an image of an A10 aircraft.

The `model-map` command can also include an optional rotational offset and scaling factor for the image, for example:

```
(setq scad (list "./data/viewpoint/Launcher" "scudb_a.flt" -90.0 1.0 0 0 0))
```

where `-90` represents the number of degrees of rotation about the z axis, `1.0` is a scaling factor that is applied to the model geometry and `0 0 0` is the X, Y, Z offset.



Rotational offset and scaling is rarely used in VR-Forces configuration files. The Model Mapper utility does not support this feature of the `model-map` command. Therefore, you cannot use it in the *ModelMap.mtl* file.

12.1.2. Defining Symbolic Names for Entity Types and Models

The basic `model-map` command includes an entity enumeration and a directory path, as in this example:

```
(model-map (list 1 1 225 1 1 0 0) (list "./data/myModels" "m1a1.flt"))
```

This format is not necessarily easy to read, particularly if you have a large number of mappings. To make the model-mapping process easier, you can define symbolic names for entity types and for model files, and use those names in the `model-map` command. For example:

```
(setq m1a1-type (list 1 1 225 1 1 0 0))  
(setq m1a1-model (list "./data/myModels" "m1a1.flt"))  
(model-map m1a1-type m1a1-model)
```

In this sequence of commands, we define `m1a1-type` as a symbolic name that represents the entity type enumeration `1 1 225 1 1 0 0`. Then we define `m1a1-model` as a symbolic name that represents the model file *m1a1.flt*. Finally, we use the symbolic names in the `model-map` command. The use of symbolic names is most beneficial when you map a particular model to more than one entity type.

12.2. Model Mapping Files Supplied with VR-Forces

VR-Forces includes the following configuration files with model mappings:

- ♦ *ModelMap.mtl* contains model-mapping commands in a format that is compatible with the Model Mapper utility.
- ♦ *aggregateMap.mtl* contains mappings for aggregate symbol icons.
- ♦ *trailMap.mtl* contains commands and mapping commands for trailing effects.
- ♦ *infantryMap.mtl* contains mapping commands for dismounted infantry models.

ModelMap.mtl, *aggregateMap.mtl*, and *trailMap.mtl* are loaded at startup by *Stealth.mtl*.

12.2.1. The *ModelMap.mtl* File

The *ModelMap.mtl* file contains all the model mapping commands for the models that VR-Forces loads by default. It uses symbolic names for mapping model paths (see [“Defining Symbolic Names for Entity Types and Models,”](#) on page 12-3.) The first portion of the file maps model paths to symbolic names. The rest of the file maps entity types to the symbolic names for the paths.

12.2.2. The *aggregateMap.mtl* File

The file *aggregateMap.mtl* provides a set of default mappings for aggregate icons. The mapping method is the same as for entity models, except that instead of the `model-map` command, you use the `agg-icon-map` command. You define a symbolic name for a model and map it to an entity type and force ID, for example:

```
;; Maps a directory path to symbolic name aggregate-path
(setq aggregate-path "../data/models/aggregate/")
;; Maps symbolic name friendly_team to the file friendly_team.flt
(setq friendly_team (list aggregate-path "friendly_team.flt"))
;; Maps entity enumeration and force ID 1 to friendly_team
(agg-icon-map (list -1 -1 -1 -1 -1 -1 -1) 1 friendly_team)
```

12.2.3. The *trailMap.mtl* File

The *trailMap.mtl* file maps trailing effects to entities. It follows the same general format as the configuration files discussed previously. You assign a symbolic name to a model file. Then you associate the symbolic name with an entity with a `trail-map` command.

The format for mapping a trail to a model is:

```
(setq trail (list "model_directory" "model_file" scale
seconds_to_dissipate percent_growth_per_second percent_overlap))
```

where:

- ♦ *trail* is the symbolic name you want to assign to this trail.
- ♦ *list* is a keyword
- ♦ *model_directory* is the relative or absolute path of the directory in which the model file is located.
- ♦ *model_file* is the FLT file that you are mapping to the symbolic name.
- ♦ *scale* specifies an x scale factor for the trail (front/back, body coordinates), a y scale factor (left/right, body coordinates), and a z scale factor (up/down, body coordinates).
- ♦ *seconds_to_dissipate* specifies the time it takes for each segment to fade out.
- ♦ *percent_growth_per_second* specifies the rate at which each trail segment gets wider, but not longer. The expansion is linear with time, so that a rate of 2.0 makes the segment twice as wide after 1 second, 3x as wide after 2, and so on.
- ♦ *percent_overlap* specifies the overlap of trail segments. If overlap is 0.0, each segment begins at the end of the last. If overlap is 0.5, the segments overlap by 50%. Negative values put spaces between the segments.

The format for the `trail-map` command is:

```
(trail-map (list enumeration) (list trail x y z))
```

where:

- ♦ *enumeration* is a seven-digit RPR FOM/DIS entity enumeration for the entity that should display the trailing effect.
- ♦ *trail* is the name of a trail specified by a `setq` command as described earlier in this section.
- ♦ *x y z* are the offsets from the center of the entity.

The following example creates a trail called VaporTrail that is associated with the model *VaporTrail.flt*. Then it maps VaporTrail to any fighter/air defense entity.

```
(setq vaporTrail (list "../data/makTrails/VaporTrail" "VaporTrail.flt" 6.0
  1.5 1.5 4.0 1.5 0.5))
(trail-map (list 1 2 -1 1 -1 -1 -1)(list VaporTrail 0.0 0 0.0))
```

You can specify more than one trail for an entity. This is useful for entities that have multiple exhausts, such as a twin engine aircraft. The following example shows two vapor trails for the entity. The X, Y, and Z coordinates specify an offset for the trails from the center of the entity.

```
(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)
)
```

12.2.4. The *modelset-2525B.mtl* Files (XR Mode Only)

The *ModelMap.mtl* file and the model settings in *params.mtl* configure the Primary Model Set. The *modelset-2525B.mtl* and *modelset-2525B-ttf.mtl* files configure alternative model sets, which uses MILSTD 2525B icons instead of 3D models to represent entities. The files are basically a shorthand version of the values specified by individual parameters in *params.mtl*. Use the comments in the file as a guide to creating other alternative model sets. If you create other model sets, include an entry for each one in *Stealth.mtl*.

12.3. Adding Your Own Model Mappings

If you want to add model mappings to the *ModelMap.mtl* file, we recommend that you use the Model Mapper tool. Use of the Model Mapper is described in Chapter 13, [Using the Model Mapper](#). You can also create custom model mapping files, either using the Model Mapper tool or a text editor. The next section explains how to create a custom model mapping file.

12.3.1. Custom Model Mapping Files

To create a model mapping file:

1. In a text file, enter `model-map` commands as described earlier in this chapter.
2. Save the file with the extension *.mtl*.
3. To load the file at startup, use the `-l` command-line option, or include a `load` command in *Stealth.mtl* (or another configuration file loaded at startup).

If you define symbolic names and map models in the same file, be sure to define the symbolic name before you use that name in a `model-map` command.



If you load your custom file through a command in *Stealth.mtl*, make sure it is loaded after (or instead of) the standard mapping files. Otherwise, your custom mappings will be overridden by the mappings in *ModelMap.mtl*.

12.4. Articulated Parts and Entity Appearances

Some MultiGen-Paradigm OpenFlight files contain information specified by the *WRM Entity Model Specification for DIS Interoperability*. This information makes it possible to depict the state of an entity's articulated parts correctly, and to switch among different entity appearances (for example, destroyed).



Although the WRM Model Specification was developed for DIS, it is also used for the HLA RPR FOM.

Specifically, these models contain information about:

- ♦ Which articulated part types are associated with which pieces of geometry.
- ♦ Which pieces of geometry correspond with which entity appearances.

For information about the WRM specification, please see Appendix E, [The WRM Specification](#).

12.5. Models Supplied with the 3D Front End

This section illustrates the entity and object models provided with 3D Front End.

Table 12-1: Air platforms



AH-1W Super Cobra



UH-60A Black Hawk



CH-46E Sea Knight



CH-53E Super Stallion



RAH-66 Comanche



OH-58 Kiowa



KA-50 Hokum



MI-24 Hind



MI-28 Havoc



MI-2 Hoplite



FA-18 Hornet



F-15E Strike Eagle



F-16 Falcon



F-117A Nighthawk



A-10 Thunderbolt

Table 12-1: Air platforms



AV-8B Harrier II



EA-6B Prowler



C-130 Hercules



SU-25 Frogfoot



MIG-29 Fulcrum



RQ-1 Predator



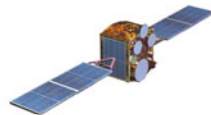
VTUAV Dragon Warrior



VTUAV Cypher



Space Shuttle Orbiter



Defense Satellite
Communications System-3

Table 12-2: Surface, Subsurface, Weapons, and Missiles



Rigid-hull inflatable boat



Landing Craft Air Cushion (LCAC)



LSD-49 Harpers Ferry class



CG-47 Ticonderoga



SSN-688 Los Angeles



CV-59 Forrestal



M4 Carbine



M136 AT4



AGM-65 Maverick



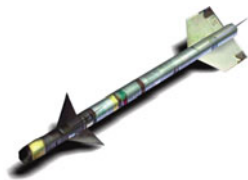
AGM-114 Hellfire



AS-12 Kegler



AGM-88 HARM



AIM-9 Sidewinder



Exocet AM-39



M-220 BGM-71 TOW Missile

Table 12-2: Surface, Subsurface, Weapons, and Missiles

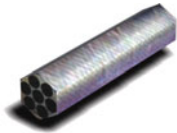
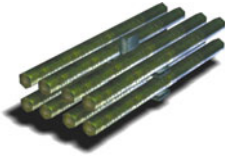
		
AIM-120 AMRAAM Slammer	MK-48 Torpedo	M260 7 Tube rocket launcher
		
M261 19 tube rocket launcher	S-8 80mm rocket system	AN-ALQ-99 tactical jamming system
		
370g Tank	MIM-104 Patriot PAC-3	SS-1C SCUD-B

Table 12-3: Land vehicles

		
M1A2 Abrams	T-80 Main Battle Tank	M110 SP Howitzer
		
MT-LB Multipurpose Armored Vehicle	M-1974 SP Howitzer	M-939A2 5-ton truck

Table 12-3: Land vehicles



T-72 Main Battle Tank



BTR-80 armored personnel carrier



M113 armored personnel carrier



BMP armored fighting vehicle



AAVC7A1 landing vehicle



M1045 HMMWV with TOW



M3 Bradley Fighting Vehicle



M58 MICLIC



M9-ACE



M-939A2 5-ton truck variant



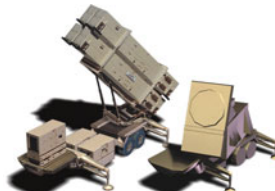
M88 Medium Recovery Vehicle



M977 HEMTT



HMMWV AVenger



MIM-104 Patriot Launcher System with radar



ZSU-23-4 Shilka

Table 12-3: Land vehicles



Table 12-4: Lifeforms and Cultural Features



Table 12-4: Lifeforms and Cultural Features



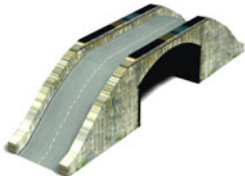
Four-story office building



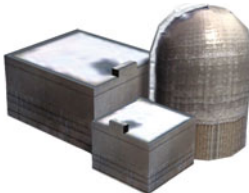
POL storage tank



Steel water tower



Arched stone bridge



Nuclear reactor



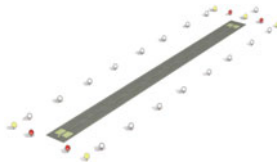
Minefield post



Concrete control tower



Metal hangar



1000 meter asphalt runway



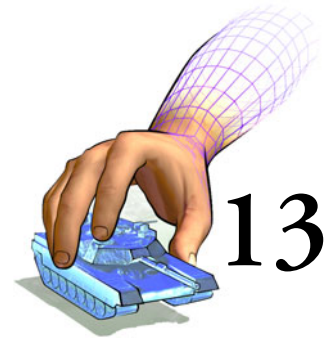
Power transmission tower



Dragon's teeth obstacle



Communications tower



Using the Model Mapper

This chapter explains how to use the Model Mapper utility to update *ModelMaps.mtl* for the 3D Front End.

Introduction to the Model Mapper	13-2
Starting the Model Mapper	13-2
The Model Mapper Interface.....	13-3
Mapping Entity Types to Model Files	13-6
Mapping Entity Types in the Entity Types Tab.....	13-7
Mapping an Entity Type to No Model	13-8
Changing a Model's Path Name.....	13-8
Specifying Textures for a Model.....	13-9
Removing a Mapping	13-9
Removing All Mappings	13-9
Saving Model Mappings	13-9
Opening Model Mapping Files.....	13-10
Entering Free Form Mappings.....	13-10
Deleting a Free Form Mapping.....	13-11
Deleting Unused Paths	13-11
Setting Model Mapper Preferences	13-11
Changing the Look and Feel of the Model Mapper	13-12
Specifying the Default Path Name	13-12
Specifying Absolute or Relative Model Paths	13-13
Specifying the 3D Preview Application.....	13-13
Turning Off the Overwrite Confirmation Prompt	13-13
Saving Preferences	13-13

13.1. Introduction to the Model Mapper

The Model Mapper is a graphical tool that helps you build model mapping files for the 3D Front End. The Model Mapper makes the process of creating model mapping files easy by allowing you to graphically associate entity types with specific model files. The Model Mapper reads existing model mapping files and writes new model mapping files or edited model mapping files. You do not have to concern yourself with the syntax of the files or worry about making typing mistakes when you edit a file.

The Model Mapper is installed in *.tools/modelmapper*. It is available only with the Windows version of the 3D Front End.

13.2. Starting the Model Mapper

The Windows version of the Model Mapper comes with a batch file, *ModelMapper.bat*, that contains the startup command.

You can start the Model Mapper on a PC in any of the following ways:

- ♦ On the Start menu, choose Programs → MÄK Technologies → VR-Forces *x.x* → Tools → VR-Forces *x.x* Model Mapper, where *x.x* is the VR-Forces release number.
- ♦ Drag a shortcut for the batch file onto the Windows desktop and start it by double-clicking.
- ♦ Double-click the batch file name in the Windows Explorer.
- ♦ Enter the pathname of the batch file in a command window or the Run window on the Start menu.

The syntax for starting the Model Mapper from the command line in Windows or Linux is:

```
java -cp "ModelMapper.jar" ModelMapper dis-fom.txt
```

13.2.1. The Model Mapper Interface

The Model Mapper has one window with two tabs (Figure 13-1). It opens with the Entity Types tab displayed.

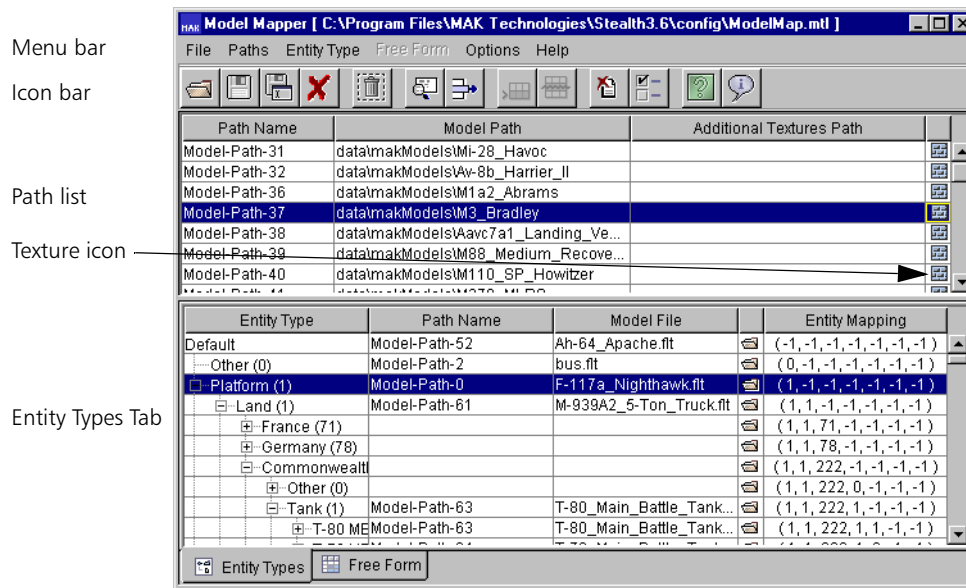


Figure 13-1. Model Mapper window, Entity Types tab

The window is divided into two sections. The Path List lists all the Path Names used in the mapping file, the model paths associated with them, and any additional texture files that you want to associate with a model. The Entity Types tab and Free Form tabs are the portion of the window in which you map model files to path names.

The column headings in the Model Mapper window have the following meanings:

- ♦ Path Name – A meaningful name (or a default name) used to identify a directory path.
- ♦ Model Path – The path of a FLT model file.
- ♦ Model File – The file name of a FLT model file.
- ♦ Additional Texture Paths – The path of texture files that you want to associate with the model. The texture files can be in one or more directories that are different from the directory that contains the model file.
- ♦ Entity Type – Text description of an HLA RPR FOM/DIS entity type.
- ♦ Entity Mapping – Enumeration of the entity type.

Table 13-1 describes the menu commands in the Model Mapper.

Table 13-1: Model Mapper menu commands

Menu	Command	Description
File	Open	Open a model mapping file.
	Save	Save the model mappings to a file.
	Save As	Save the model mappings under a new name.
	Exit	Close the Model Mapper.
Paths	Delete Unused	Delete all paths that do not have a model mapped to them.
Entity Type	Show Mapped Types	Expand the Entity Type tree to show all mapped entity types.
	Clear Selected	Removes a mapping to an entity type.
Free Form	Add	Adds a new free form mapping.
	Delete Selected	Deletes a free form mapping.
Options	Clear All Mappings	Removes all mappings from both tabs.
	Preferences	Allows you to set preferences for how the Model Mapper looks and behaves.
Help	Help on Model Mapper	Opens online help. (Online help for the 3D Front End also includes help for the Model Mapper.)
	About	Displays release information.

The Model Mapper Toolbars

The toolbar provides icons for most menu commands. Icons have tool tip help. If you hold the mouse cursor over the icon for several seconds, a small window is displayed, which identifies the function of the icon. Some icons are available only when a particular tab is selected. [Figure 13-2](#) identifies the icons. You can dock the toolbar to any side of the Model Mapper window and you can drag it off the window to be free-floating.

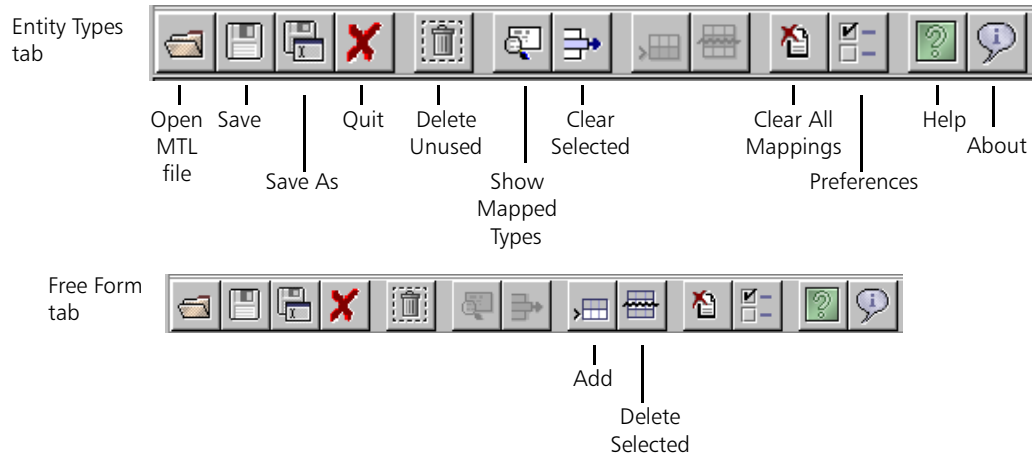


Figure 13-2. Model Mapper toolbar

Moving the Toolbar

To move the toolbar:

1. Hold the mouse cursor over the background of the toolbar.
2. Press and hold the left mouse button and drag the mouse to any side of the window. A gray shadow of the toolbar follows the cursor.
3. To dock the toolbar to a side of the Model Mapper window, move the shadow until it orients itself to the side of the window where you want to place it, and is bordered in orange. Then release the mouse button.

To use the toolbar as a free-floating window, release the mouse button while the shadow has a black border.

Docking A Floating Toolbar to the Model Mapper Window

- To dock a floating toolbar to the Model Mapper window, close the toolbar window. (Click the X in the upper right corner of the window.)

13.3. Mapping Entity Types to Model Files

The Entity Types tab lets you map model files to a set list of Entity Types. The Free Form tab (Figure 13-3) lets you input any entity mapping (entity type enumeration) that you want and associate a model file with it.

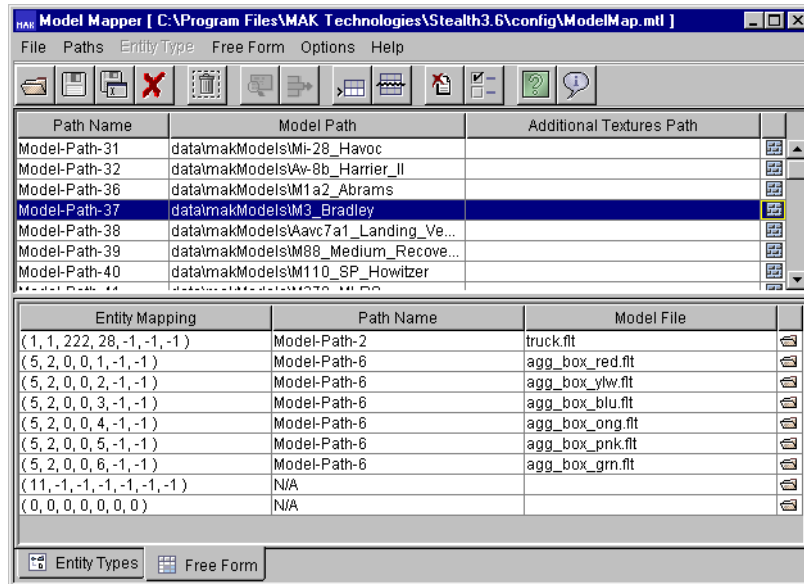


Figure 13-3. Free Form tab

13.3.1. Mapping Entity Types in the Entity Types Tab

When you start the Model Mapper it loads the Entity Types that are part of the HLA RPR FOM and DIS. They are listed in the Entity Type column. The enumeration for each entity type is listed in the Entity Mapping column.

To map a model file to an entity type:

1. In the list of entity types, locate the entity that you want to map.
2. Click the folder icon on the same line as the entity type. The Select OpenFlight File dialog box opens.
3. In the directory window, select the FLT file that you want to map to the entity.
4. Click Select.

When you select a FLT file, the following information is displayed:

- ♦ The name of the file is entered in the Model File cell.
- ♦ A Path Name is assigned to the file.
- ♦ The full path of the file is displayed in the Model Path column of the Path List
- ♦ The Path Name is displayed in the Path Name column of the Path List.

As you add new mappings, the information is displayed in the Path List.



If you make any change to a model mapper file other than changing the pathnames from relative to absolute, an asterisk (*) is appended to the file name in the title bar. When you save the file, the asterisk is removed.

13.3.2. Mapping an Entity Type to No Model

There are times you may want to map an entity type to no model, making it invisible. Figure 13-4 illustrates mapping an entity to no model.

Entity Mapping	Path Name	Model File	
(1, 1, 222, 28, -1, -1, -1)	Model-Path-9	sa6.flit	
(5, 2, 0, 0, 1, -1, -1)	Model-Path-6	agg_box_red.flit	
(5, 2, 0, 0, 2, -1, -1)	Model-Path-6	agg_box_ylw.flit	
(5, 2, 0, 0, 3, -1, -1)	Model-Path-6	agg_box_blu.flit	
(5, 2, 0, 0, 4, -1, -1)	Model-Path-6	agg_box_ong.flit	
(5, 2, 0, 0, 5, -1, -1)	Model-Path-6	agg_box_pnk.flit	
(5, 2, 0, 0, 6, -1, -1)	Model-Path-6	agg_box_grn.flit	
(11, -1, -1, -1, -1, -1, -1)	N/A		
(0, 0, 0, 0, 0, 0, 0)	N/A		

 Entity Types
  Free Form

Figure 13-4. Mapping an entity to no model

To map an entity type to no model:

1. In the Path Name column, enter N/A.
2. Leave the Model File column blank.

13.3.3. Changing a Model's Path Name

Pathnames are assigned in the form Model-Path-*n*, where *n* is an integer that gets incremented with each additional model path added. You can change the default model pathname to something more descriptive if you want to.

To change the Path Name:

1. In the Path List, double-click the Path Name that you want to edit. The cell becomes editable.
2. Change the name to whatever you want. You can enter a fully qualified path. When you next open the file, the Model Mapper recognizes that the path name is a path, puts it in the Model Path column, and assigns a default Path Name to it.
3. Press **ENTER**.

13.3.4. Specifying Textures for a Model

You can specify one or more directories that contain texture files to be used for a model.

To specify a texture directory:

1. In the Path List, click the texture icon to the far right of the path for which you want to specify textures. The Select Texture Path to Add dialog box opens.
2. Navigate the directory hierarchy by double-clicking directories that you want to open.
3. Select the directory that contains the texture files.
4. Click Select.



To select textures, you do not select individual texture files. You only select the directory in which the files are located.

13.3.5. Removing a Mapping

You cannot delete an entry in the Entity Type list, but you can remove a mapping.

To remove a mapping:

1. Select the row that has the mapping you want to remove.
2. Choose Entity Type → Clear Selected.

13.3.6. Removing All Mappings

You can remove all mapping from both tabs.

- To clear all mappings, choose Options → Clear All Mappings.

13.3.7. Saving Model Mappings

When you are finished mapping FLT files to entity types, save the mappings to a configuration file. You can read the file back into the Model Mapper at a later time to edit it.

To save the mappings to a configuration file:

1. Choose File → Save. The Write MTL File dialog box opens.
2. Specify a directory and file name to use.
3. Click Write.

13.3.8. Opening Model Mapping Files

You can open a model mapping file in the Model Mapper and edit it.

To open a model mapping file:

1. Choose File → Open. The Open MTL File dialog box opens.
2. Select a configuration file to open.
3. Click Open.



The Model Mapper will try to open any configuration file that you select. However, if a configuration file has not been created by the Model Mapper it will not be able to read it, so be certain that the file you are opening is a configuration file created by the Model Mapper.

13.3.9. Entering Free Form Mappings

The mappings on the Entity Types tab are based on the HLA RPR FOM/DIS enumerations. On the Free Form tab, you can enter any mapping that you like, for example, mappings for new enumeration extensions.

To enter a free form mapping:

1. With the Free Form tab selected, choose Free Form → Add. A new line is added to the tab. The Entity Mapping column contains a string of -1s.
2. Click the entity mapping. The Edit Entity Mapping dialog box opens ([Figure 13-5](#)).

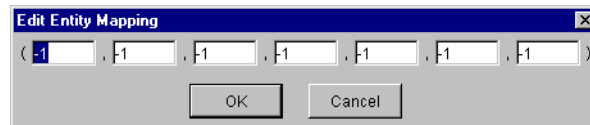


Figure 13-5. Edit Entity Mapping dialog box

3. Edit the entity mapping.
4. Click OK.
5. Specify a path name.
6. Specify a model file.

13.3.10. Deleting a Free Form Mapping

To delete a mapping in the Free Form tab:

1. Select the mapping you want to delete.
2. Choose Free Form → Delete Selected.

13.3.11. Deleting Unused Paths

As you update and edit model mappings, the path list may contain paths that you are no longer using. To clean up the list, you can remove unused paths.

- To remove unused paths from the path list, choose Paths → Delete Unused.

13.4. Setting Model Mapper Preferences

You can set the following preferences:

- ♦ The look and feel of the Model Mapper window
- ♦ The default path name prefix
- ♦ Use of absolute or relative model paths
- ♦ The 3D previewer for the model mappings.

You can save the preferences to file for use the next time you use the Model Mapper.

13.4.1. Changing the Look and Feel of the Model Mapper

The look and feel of the Model Mapper window defaults to the standard look and feel for the platform on which it is running. However, you can change the look and feel to any of the following:

- ♦ MS-Windows
- ♦ Metal (Java)
- ♦ CDE/Motif.

To change the look and feel:

1. Choose Options → Preferences. The Preferences dialog box opens (Figure 13-6).

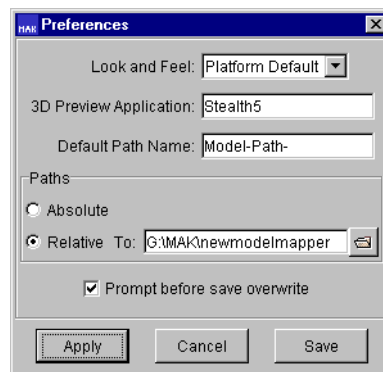


Figure 13-6. Preferences dialog box

2. Select an option on the Look and Feel list.
3. Click Apply.

13.4.2. Specifying the Default Path Name

The Model Mapper uses symbolic names to specify paths in model-map commands. By default, the symbolic names are constructed by joining the text Model-Path- with an integer.

Changing the default path name does not change any of the existing symbolic names. It affects all path names created after you change your preference.

To change the default text for the symbolic name:

1. Choose Options → Preferences. The Preferences dialog box opens (Figure 13-6).
2. In the Default Path Name box, enter the text you want to use.

13.4.3. Specifying Absolute or Relative Model Paths

You can specify that the pathnames in model mapper files be absolute or relative to a specified directory.

To specify the pathname style:

1. Choose Options → Preferences. The Preferences dialog box opens (Figure 13-6).
2. In the Paths box, select the Absolute or Relative option.
3. If you choose the Relative option, you can specify the directory to which paths are relative.



Do not switch back and forth between relative and absolute paths without first closing the model map file and then reopening it in the Model Mapper. Switching back and forth in the same session may result in inaccurate path names.

13.4.4. Specifying the 3D Preview Application

To specify the 3D preview application:

1. Choose Options → Preferences. The Preferences dialog box opens (Figure 13-6).
2. Enter the path of the preview application in the 3D Preview Application field.
3. Click Apply.

13.4.5. Turning Off the Overwrite Confirmation Prompt

To turn off the overwrite confirmation prompt when saving preferences:

1. Open the Preferences dialog box.
2. Clear the Prompt Before Save Overwrite box.

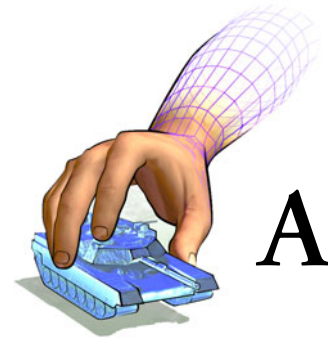


When you save this changed preference, you are prompted to confirm the overwrite. Thereafter, you will not be prompted.

13.4.6. Saving Preferences

To save your preferences for the next Model Mapper session:

1. Choose Options → Preferences. The Preferences dialog box opens.
2. Click Save.



Object Parameters

This appendix describes the high-level parameters in the object parameter database. The OPD Editor has context-sensitive help for most parameters and also describes them in the online help system. It also has reference material for local and remote subcomponents and object type enumerations.

The Structure of an Entity Parameter	A-2
The Parameter Type Hierarchy.....	A-3
Vrf Base Object Parameters.....	A-3
Vrf Object Parameters	A-7
Moving Object Parameters	A-8
Parameter Types for Entity Platforms	A-8
Entity Parameters.....	A-8
Object Type Enumerations	A-9
Object Super-type	A-9
AggregateKind.....	A-9
ObjectKind	A-10
Aggregate Category	A-10
Aggregate Specific.....	A-11
Point Object Category.....	A-12
LinearObjectKind	A-12
LinearObjectCategory	A-12
Interaction.....	A-12

A.1. The Structure of an Entity Parameter

The object parameter database uses MÄK Technologies LISP (MTL). (For details about MTL, please see “[MÄK Technologies Lisp \(MTL\) Files](#),” on page 2-2.) A parameter block has the form:

```
(block_name
  ( parameter_name value)
  ...
  (parameter_name
    (sub-parameter value)
  )
  and so on
)
```

Entries have the following requirements:

- ♦ The block_name can be anything that you like.
- ♦ 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)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 current 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.

A.2. The Parameter Type Hierarchy

Figure 7-3, in “Parameter Types,” on page 7-9, illustrates the hierarchy of parameter types. In this section, we describe the parameters associated with each **parameter-type**.

A.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). Please see the General Parameters topic in the OPD Editor online help for parameter descriptions. In addition to parameters, each object has an object type and parameter type, as described in [Table A-1](#).

Table A-1: Object and parameter type

Parameter	Description
object-type	8-digit object type. Not actually a parameter. Specifies the key used to look up the entry. (Required) For more information, please see “Object Types,” on page 7-6.
parameter-type	One of the following (string): ♦ "vrf-base-object-param" ♦ "vrf-object-param" ♦ "moving-object-param" ♦ "fixed-wing-param" ♦ "ground-vehicle-param" ♦ "rotary-wing-entity-param" ♦ "missile-param" ♦ "surface-entity-param" ♦ "human-param" ♦ "cultural-feature-param" (Required) The value must be in quotation marks.

Local Objects 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 A-2](#) lists the subcomponent parameters and their possible values.

Table A-2: Local-objects subcomponent parameters (Sheet 1 of 2)

Subcomponent parameter	Description
state-repository	(Required) Maintains the state information for the object. An object that has components (sensors, controllers, and actuators) for simulation will set and retrieve information in the repository. Some simulation components expect a specific kind of state repository. In general, you will want to specify the state repository type that most closely matches the kind of entity you expect to simulate. May be one of the following 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 entity-specific state repository types that are derived from this type, this kind of repository is not typically specified. ♦ "vrf-overlay-object-state-repository": For graphical objects. ♦ "vrf-moving-object-state-repository": For any object that you expect to simulate movement. Pseudo-aggregate entities have this kind of state repository. The more entity-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. ♦ "human-state-repository": For lifeform entities. ♦ "cultural-feature-state-repository": For cultural features that are created as entities.

Table A-2: Local-objects subcomponent parameters (Sheet 2 of 2)

Subcomponent parameter	Description
network-interface	(Optional) The network interface handles the publication of the object over the network. It acts as an interface between an entity's state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of entity you are simulating. May be one of the following types: ♦ "aggregate-local-entity-net-interface": For aggregate entities. ♦ "pseudo-aggregate-local-entity-net-interface": For aggregate entities that have subordinate entities that are separately simulated. ♦ "individual-local-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. ♦ "local-environmental-net-interface": For entities that are best represented as an environmental process. Non-entity objects such as control objects (waypoints, routes, areas) 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 an entity'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 $\text{rand}(\text{net-interface-min-tick-period}/2)$
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 entities). ♦ "component-manager"
plan-manager	(Optional) The entity plan manager. If you want an entity 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. ♦ "local-task-manager": For individual entities or true aggregates.

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 A-3](#) lists the subcomponent parameters and their possible values.

Table A-3: Remote object subcomponent parameters (Sheet 1 of 2)

Subcomponent parameter	Description
<code>state-repository</code>	(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 <code>"vrf-moving-object-state-repository"</code> is sufficient. A more specific state repository may be specified. However, some of the entity-specific data may not be set. May be one of the following types: ♦ <code>"vrf-object-state-repository"</code>: Derived from <code>vrf-object-base-state-repository</code>, it adds the concepts of resources, sector of responsibility, engagement rules, and targeting. Because there are more entity-specific state repository types that are derived from this type, this kind of repository is not typically specified. ♦ <code>"vrf-overlay-object-state-repository"</code>: For graphical objects. ♦ <code>"vrf-moving-object-state-repository"</code>: For any object that you expect to simulate movement, including individual and aggregate entities. ♦ <code>"ground-entity-state-repository"</code>: For ground vehicles, such as tanks, trucks, dismounted infantry. ♦ <code>"rotary-wing-entity-state-repository"</code>: For rotary-wing aircraft. ♦ <code>"fixed-wing-entity-state-repository"</code>: For fixed-wing aircraft. ♦ <code>"missile-entity-state-repository"</code>: For missile and rocket entities. ♦ <code>"human-state-repository"</code>: For lifeform entities. ♦ <code>"cultural-feature-state-repository"</code>: For cultural features that are created as entities. ♦ <code>"surface-entity-state-repository"</code>: For surface entities.

Table A-3: Remote object subcomponent parameters (Sheet 2 of 2)

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 an entities state repository and its network (VR-Link) representation. Choose one that most closely matches the kind of entity you are simulating. May be one of the following types: • "vrf- aggregate -remote-entity-net-interface": For remote aggregate entities. • "vrf-individual-remote-entity-net-interface": For individual platform entities, such as tanks, aircraft, dismounted infantry. • "vrf-remote-environmental-net-interface": For entities that are best represented as an environmental process. Non-entity objects such as control objects (waypoints, routes, areas) 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 an entity'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.)

A.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 aggregates and entities) 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**.

A.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 individual entities and aggregates. Parameters include support points, maximum speed, maximum reverse speed, turning radius, maximum slope, ordered speed, maximum acceleration, maximum deceleration, and fuel efficiency. Pseudo-aggregates are moving objects and have this parameter type. All of the platform-specific parameter types derive from this one.

A.2.4. Parameter Types for Entity Platforms

[Table A-4](#) lists the parameter types used by each entity type.

Table A-4: Parameter types for entities

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
Cultural feature	cultural-feature-param	moving-object-param
Missile	missile-param	moving-object-param
Pseudo-aggregate	moving-object-param	vrf-object-param

A.3. Entity Parameters

For descriptions of entity parameters, please see the descriptions of individual components in the OPD Editor online help.

A.4. Object Type Enumerations

This section lists the object type enumerations as listed in the header file *objTypeEnum.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.

A.4.1. Object Super-type

The header file *objTypeEnum.h* contains the VR-Forces extensions to the DIS/RPR FOM enumerations. The enumerations specify individual, aggregate, and non-entity 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 entity types are not guaranteed to be unique across all of the possible individual and aggregate types, the extra enumerated field provides a way to uniquely identify all objects in a VR-Forces simulation.

Other	0
Individual	1
Aggregate	2
PseudoAggregate	3
TrueAggregate	4
Interaction	5

A.4.2. AggregateKind

The enumeration for *DtAggregateKind* should start with 0, but VR-Forces starts with 10, to ensure that entity types will be unique.

Other	10
MilitaryHierarchy	11
CommonType	12
CommonMission	13
SimilarCapabilities	14
CommonLocation	15
DtSimArtPartKind	20

A.4.3. ObjectKind

Extension of entity kind to represent non-entity objects, such as control objects (control points, routes, phase lines, obstacles, etc.)

DtPointObjectKind	16
DtLinearObjectKind	17
DtArealObjectKind	18
DtCircleObjectKind	19
DtTextObjectKind	20

A.4.4. Aggregate Category

VR-Forces extends the enumeration to include *DtAggregateCategoryForce*, which represents the force level. All of the entries from Force through the end of the list, 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

Aggregate Subcategory

The Subcategory field enumerations for Kind field = Military Hierarchy specify the Echelon Type of the aggregate. Not all of the echelon types are applicable to each of the categories.

In the following 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		

A.4.5. Aggregate Specific

The Specific field enumerations for Kind field = Military Hierarchy specify whether the aggregate unit contains a headquarters, and is enumerated as follows.

Noheadquarters	0
ContainsHeadquarters	1

A.4.6. Point Object Category

Subcategories for use with the Kind field = *DtPointObjectKind*.

Other	0
ControlPoint	1
TargetPoint	2
Marker	3

A.4.7. LinearObjectKind

Subcategories for use with the Kind field = *DtLinearObjectKind*.

Other	0
PhaseLine	1
Route	2

A.4.8. LinearObjectCategory

Subcategories for use with the Kind field = *DtArealObjectKind*.

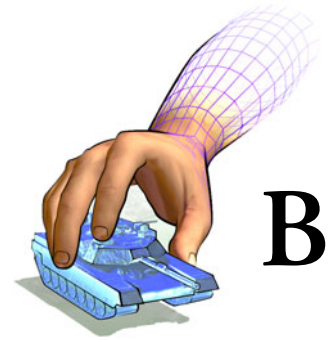
Other	0
ControlArea	1
Obstacle	2

ArealObjectCategory

A.4.9. Interaction

Kinds for Supertype interaction.

Other	0
Fire	1
Detonate	2



Editing the Create Menu

This chapter describes how to edit the Create menu manually.

The Create Menu.....	B-2
The vrf.ent Configuration File	B-2
Object Blocks in vrf.ent.....	B-4
Reorganizing the Create Menu	B-4
Editing Menu Items	B-4
Adding New Entity Types to VR-Forces.....	B-5
Adding a New Entity or Control Object to vrf.ent	B-5
Editing the Overlay Object Menus.....	B-5

B.1. The Create Menu

By default, the Create menu is generated automatically from the *entity.mst* file. However, you can use a customized Create menu if you want to. The recommended way to create a customized Create menu is to use the Entity Editor, as described in “Editing the Create Menu,” on page 5-19. This appendix explains how to edit the Create menu in a text editor.

You can change the Create menu by:

- ♦ Reorganizing entries
- ♦ Editing existing entries
- ♦ Adding new entries.

You modify the Create menu and its sub-menus by editing the following files:

- ♦ *vrf.ent* – configures entities, aggregates, and control objects
- ♦ *overlayMenu.mtl* – configures the overlay object sub-menus.

If you make changes to *vrf.ent*, you may also need to make changes to the object parameter database.

B.2. The *vrf.ent* Configuration File

The *vrf.ent* file contains the information VR-Forces uses to build the Entity and Aggregates menu options on the Create menu, and the list of control objects. The menu structure is completely configurable. It consists of a set of nested menu blocks. The highest block specifies the name of the menu. The second level specifies entries on the main menu. Nested menu blocks specify entries for submenus. The `create-entity-menu-item` entries specify the objects you can create. [Figure B-1](#) is a highly edited version of the *vrf.ent* file that shows nested menus and an illustration of the menu structure that would be created by the sample *vrf.ent* file.



Figure B-1. Abbreviated *vrf.ent* file and menus created

B.2.1. Object Blocks in *vrf.ent*

The entry for each type of entity that you can create defines a label, object type, and force type, similar to the following:

```
(create-entity-menu-item
  (label "M1A2 Tank")
  (object-type 1 (1 1 225 1 1 3 0))
  (force-type ForceFriendly)
)
```

where:

- ♦ **label** specifies the text that is displayed on the menu.
- ♦ **object-type** is the eight-digit object type for the object. It must match an object-type in the object parameter database using exact match or best match criteria.
- ♦ **force-type** determines the force assigned to a newly created object. The options are:
 - ForceOther
 - ForceNeutral
 - ForceFriendly
 - ForceOpposing
 - ForceUserDefined5 up to ForceUserDefined255.

B.2.2. Reorganizing the Create Menu

The default *vrf.ent* file organizes entities by force and domain. It organizes aggregates by hierarchical level. If you would prefer to organize entities differently, all you have to do is create menu blocks with your preferred organization and move the appropriate `create-entity-menu-item` blocks to the new menu blocks. For example, you could move the control object entries to a Control Objects sub-menu.

B.2.3. Editing Menu Items

You can change the labels and force types for entries on the Create Menu by editing the text in *vrf.ent*. You can also change the objects associated with the entries by editing the **object-type**. However, if you change an object type, you must be sure that it matches an entry in the object parameter database. Please see [“Adding New Entity Types to VR-Forces,”](#) on page B-5 for details.

B.3. Adding New Entity Types to VR-Forces

You can add new entity types to VR-Forces without using the VR-Forces API toolkit.

To add a new entity type, you must:

1. Add the entity to the Create menu.
2. Ensure that the object parameter database has an entry whose object-type matches the object type in *vrf.ent* or meets best-match criteria.

B.3.1. Adding a New Entity or Control Object to *vrf.ent*



Before making changes to a configuration file, be sure to make a backup copy of the original file.

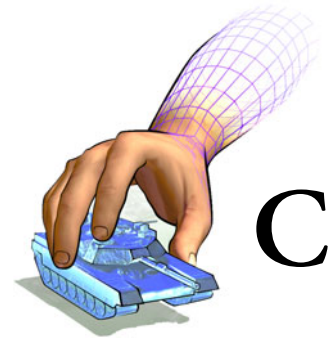
To add a new object to *vrf.ent*:

1. Open *vrf.ent* in a text editor.
2. To add an entity (including preconfigured aggregates), add a new **create-entity-menu-item** block to the file. To add a control object, add a new **create-object-menu-item**. Be sure that you match the syntax and parentheses of an existing block. (You could copy a block, paste it at the end of the existing block, and change the entries for the various fields as described in the following steps.)
3. In the **label** field, enter a string for the new entity. This is the text that will be displayed on the menu.
4. In the **object-type** field, enter an 8-digit object-type for the entity. Do not use wildcard values. The object type must match (using the best match method) an object type that is specified in *vrfSim.opd* (or an alternative OPD file.)
5. In the **force-type** field, enter the force type for the entity.
6. Save the file.

The next time you start VR-Forces, the new entity will be listed on the appropriate submenu of the Create menu.

B.4. Editing the Overlay Object Menus

The layout of the Overlay Objects submenus is configured in the *overlayMenu.mtl* file. For details, please see [“Configuring Overlay Objects,”](#) on page 2-22.



Hazard Cloud Message Formats

This chapter describes how to edit the Create menu manually.

Hazard Cloud Environmental Process Messages	C-2

C.1. Hazard Cloud Environmental Process Messages

The 3D Front End supports display of chemical liquids, chemical vapor clouds, and biological clouds. The types for the supported records are:

```
BoundingBoxVolume = 65536; // 00010000 (H)
Gaussina Puff = 1879048192; // 70000000 (H)
Uniform Box = 2147483647; //7FFFFFFF (H)
Biological Vapor = 4096; // 00001000 (H) // Bio
Chemical Vapor = 4097; // 00001001 (H) // Chemical Vapor
Chemical Liquid = 4099; // 00001003 (H) // Chemical Liquid
```

The 3D Front End updates cloud location each frame based on the velocity. Sigmas are updated each frame based on delta sigmas.

For each cloud, you can send a bounding sphere, a geometry type for display (either a gaussian puff or a uniform box), and a state record. The state record's geometry index points to the geometry record to use to visualize it. The bounding sphere is optional. An environmental process message can contain multiple clouds.

The following tables describe the data format of the environmental process messages for these records.

Table C-1: Bounding Sphere Record (EBV DOC 2003, Section 12.3.4.5)

Parameter	Value
Type	Unsigned 32
Length	Unsigned 32
Index	Unsigned 8
Padding	Unsigned 8
Location	X - Float 64 Y - Float 64 Z - Float 64
Radius	Float 32
Padding	Float 32

Table C-2: Gaussian Puff Record (EBV DOC 2003, Section 12.3.4.15)

Parameter	Value
Type	Unsigned 32
Length	U16
Index	U8
Padding	U8
Location	X - Float 64 Y - Float 64 Z - Float 64
Originating Location	X - Float 64 Y - Float 64 Z - Float 64
Sigma X	Float 32
Sigma Y	Float 32
Sigma Z	Float 32
d(Sigma X)/dt	Float 32
d(Sigma Y)/dt	Float 32
d(Sigma Z)/dt	Float 32
Orientation (radians)	Psi - Float 32 Theta - Float 32 Phi - Float 32
Velocity	X - Float 32 Y - Float 32 Z - Float 32
Angular Velocity	About X-axis - Float 32 About Y-axis - Float 32 About Z-axis - Float 32
Centroid Height	Float 32

Table C-3: Uniform Box Record

Parameter	Value
Type	U32
Length	U16
Index	U8
Padding	U8
Location	X - Float 64 Y - Float 64 Z - Float 64
Linear Velocity	X - Float 32 Y - Float 32 Z - Float 32
X scale	Float 32
Y scale	Float 32
Z scale	Float 32
Orientation (radians)	Psi - Float 32 Theta - Float 32 Phi - Float 32
Angular Velocity	About X-axis - Float 32 About Y-axis - Float 32 About Z-axis - Float 32

Table C-4: Chemical State Record

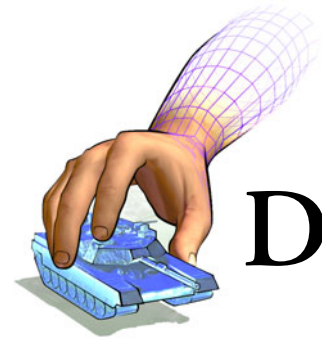
Parameter	Value
Type	U32
Length	U16
Index	U8
Padding	U8
Chemical Agent	U16
Geometry Index	U16
Agent Mass	Float 32
MMD	Float 32
Probability	Float 32

Table C-5: Chemical Liquid Record

Parameter	Value
Type	U32
Length	U16
Index	U8
Padding	U8
Chemical Agent	U16
Geometry Index	U16
Agent Mass	Float 32
Minimum Droplet Size	Float 32
Maximum Droplet Size	Float 32
Average Droplet Mass	Float 32
Purity	Float 32
Reserved	Float 32
Probability	Float 32

Table C-6: Biological State Record

Parameter	Value
Type	U32
Length	U16
Index	U8
Padding	U8
Biological Agent	U16
Geometry Index	U16
Agent Mass	Float 32
Minimum Particle Size	Float 32
Maximum Particle Size	Float 32
Average Particle Mass	Float 32
Purity	Float 32
Viability	Float 32
Probability	Float 32



The UTM Coordinate System

This appendix describes six commonly used forms of Universal Transverse Mercator (UTM) coordinates.

Standard Definition of the UTM	D-2
Arbitrary Offset UTM	D-2
Standard Offset UTM.....	D-3
Non-Offset UTM	D-3
USGS or Civilian Mapping UTM	D-3
Milgrid UTM	D-3
ModSAF UTM.....	D-3

(Reference: Map Projections - A Working Manual By John P. Snyder, U. S. Geological Survey Professional Paper 1395)

D.1. Standard Definition of the UTM

All databases that ship with the 3D Front End use Universal Transverse Mercator (UTM) coordinates. There appear to be at least six commonly used forms of UTM. Although they may be called something different than the names given them in this appendix, they have the described characteristics.

In general, in UTM, 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.

The VR-Forces uses the following 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), the 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.

D.2. Arbitrary Offset UTM

MÄK products usually use the arbitrary offset UTM.

Arbitrary offset UTM does not use the zone directly and does not use the quadrangles at all. The origin is specified via any latitude/longitude position. Internally, calculations are performed with respect to the non-offset UTM origin of the zone and then expressed as locations with respect to the arbitrary offset origin. In other words, locations are offsets from the arbitrary origin. The non-offset UTM or standard offset UTM location could be determined by calculating the arbitrary offset origin in these systems (that is, its zone, and so on) and adding on the arbitrary offset location. This system is almost always used for database generation because it allows greater precision since the offsets from the origin are typically smaller than from a standard or non-offset origin.

D.3. Standard Offset UTM

The standard offset UTM removes all negative coordinates. It is similar to non-offset UTM in that the northing portion of the origin is at the equator. It differs in that the easting portion of the origin is actually 500K meters west of the central meridian, that is, outside of the longitudinal strip. South of the equator, 10M meters are added, so a quadrangle letter designator is sometimes necessary to distinguish north and south. Numbers increase to the north and east.

D.4. Non-Offset UTM

The origin of the zone is at the central meridian (in the middle of the longitude strip) at the equator, that is, there are three degrees longitude within the zone to either side of the origin.

D.5. USGS or Civilian Mapping UTM

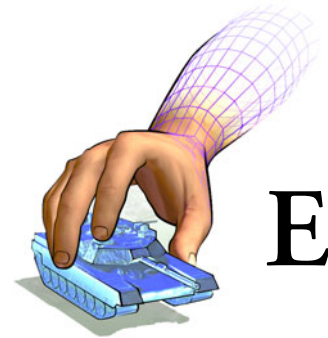
The USGS or civilian mapping UTM is essentially a non-quadrangle standard offset UTM. It only uses a zone number and an X and Y coordinate to define a point and the ellipsoid and hemisphere are known or understood. This system could not be used to describe points in both hemispheres.

D.6. Milgrid UTM

This is essentially a standard offset UTM which uses the quadrangles (quadrangle letter designator is needed) and in addition, further subdivides the quadrangles into grid squares 100K meters on a side, which are designated by double letters.

D.7. ModSAF UTM

ModSAF UTM appears to be a modified USGS civilian mapping UTM. The modification is that ModSAF reintroduces negative numbers to describe coordinates in the southern hemisphere.



The WRM Specification

Some MultiGen OpenFlight files contain certain information specified by the *WRM Entity Model Specification for DIS Interoperability*. This information makes it possible for the 3D Front End to depict the state of an entity's articulated and attached parts correctly, and to switch among different entity appearances.

This appendix contains information that can help you build entity models that comply with the WRM specification, or add that compliance to existing models.

The WRM specification was developed by MÄK Technologies, Paradigm Simulation Inc., MultiGen Inc., and SAIC, for the ARPA War Breaker project.

i

Although the WRM specification was developed for DIS, it is used in the HLA RPR FOM.

Background.....	E-2
General Information	E-3
Articulated Parts.....	E-4
Entity Appearances	E-5
Appearance Attributes Supported by 3D Front End Models.....	E-6
Attached Parts.....	E-8
Building Models that Can Have Attached Parts.....	E-8
Building Models that Can Be Attached Parts.....	E-8
Animation.....	E-9
WRM Model Examples	E-9

E.1. Background

In order for the 3D Front End to be able to change the state of a model based on incoming information, certain information about the model must be available to it. For example, which piece of the model's geometry is associated with which articulated part, and which piece of a model is associated with a particular entity appearance attribute.

This appendix describes a method of encoding information within an OpenFlight format model, so that the 3D Front End can make these associations. For the most part, information is associated with a MultiGen bead by including it as a comment associated with the bead. Comments that contain information that is meant to be interpreted by the 3D Front End begin with the token, `@dis`.

The 3D Front End depicts the movement of articulated parts and changes in entity appearances only when using models constructed to this specification. The 3D Front End also places attached parts at a specified attachment station when the attaching and attached models have been constructed to this specification.

This appendix represents a subset of the Model Specification for DIS Interoperability (version 001). For more details on any of the sections of this file, please refer to that document.

Although the 3D Front End implements only a subset of the functionality enabled by the Model Specification, we recommend that models conform to as much of the Specification as is appropriate for the model, to promote compatibility with future versions of the 3D Front End, and with visualization applications from other vendors.

E.2. General Information

This section lists a series of points for meeting the WRM specification:

- ♦ The origin of the model's coordinate system should be located as follows, depending on the domain of the entity:
 - Land: on base of vehicle, on axis of yaw rotation.
 - Surface: On axis of center of buoyancy, in plane of waterline for unloaded ship.
 - Subsurface: On axis of center of buoyancy, in plane of waterline for surfaced submarine.
 - Air and Space: At center of gravity.

- ♦ The model's orientation should be such that the positive X axis points through the right side of the vehicle, the positive Y axis points through the front of the model, and the positive Z axis points through the top of the model. This is the normal orientation of OpenFlight models.

- ♦ The top-level node of the whole model should have the following two-line comment:

```
@dis version_number spec_version  
@dis entity_type entity_type
```

for example,

```
@dis version_number 1  
@dis entity_type 1:1:225:1:1:1:0
```

The first line indicates that the model conforms with version 1 of the model specification. The second line indicates that this model represents an M1A1 Abrams tank, since 1:1:225:1:1:1:0 is the enumeration for that entity type.

♦



The 3D Front End currently ignores the entity type line in the comment, since it uses the configuration files for model mapping.

- ♦ Remember that all comments will be parsed by the 3D Front End, so spelling counts.

E.3. Articulated Parts

In order for the 3D Front End to properly depict the state of articulated parts, models must conform to the following specifications:

- There should be one DOF bead for each articulated part. (This means that if there are several LOD's, they should be children of the DOF, and not vice-versa.) Each articulation DOF should be a descendant of the DOF for the part to which it is attached (unless of course the parent part is just the base model itself.) For example, the DOF for a gun should be a descendant of the DOF for a turret.
- Each articulated part DOF bead should have all of the geometry and other beads associated with the part underneath it. This DOF node should have a comment associated with it. The comment should be of the form:

```
@dis articulated_part part-number
```

where *part-number* is the enumeration for the particular type of part that this part of the model represents. For example, the part number of a turret is 4096, and the part number of a gun is 4416.

Therefore the child group of a turret's DOF should be commented with:

```
@dis articulated_part 4096
```

The child group of a gun's DOF should be commented with:

```
@dis articulated_part 4416
```

- The WRM Model Specification specifies that information about the legal motions of each articulated part be encoded in the comment field. Currently, the 3D Front End does not use this information.
- Do not include static offsets in the DOF attributes. That is, when you set all of the DOF parameters (X, Y, Z, heading, pitch, roll) to 0, the model should appear in its neutral position.

E.4. Entity Appearances

The 3D Front End supports switching between different pieces of a model's geometry based on appearance attributes. For instance, a tank's turret should look different if the hatch is open or closed. An entire tank should look different if it is damaged or destroyed.

Switch beads are used to represent a choice between several different children, based on a particular appearance attribute.

We define a switch bead as a group bead with a comment of the form:

```
@dis switch attribute
```

where *attribute* is the name of the attribute which should be used to select a child. (For example, damage, smoke, hatch, and so on.) [“Appearance Attributes Supported by 3D Front End Models,”](#) on page E-6, lists the attributes you can use with the 3D Front End.

The children of the switch bead represent the different portions of a model among which the 3D Front End will select, based on the appearance attribute. If a switch bead is commented with:

```
@dis switch hatch
```

it may, for example, have three children containing geometry for a closed hatch, popped hatch, and open hatch.

These children of the switch bead must be commented so that the 3D Front End knows which child matches which values of the appearance attribute. The comment is of the form:

```
@dis state value-list
```

value-list is of the form $m[,m]^+$ where m is one of the following:

- ♦ An integer representing the enumeration of a value that an appearance attribute may take on, for example, 1 for hatch closed.
- ♦ Two integers, a and b , separated by a hyphen, representing all enumerations from a through b , for example, 2-5 for all states whose enumerations are 2, 3, 4, or 5.
- ♦ The integer -1, which indicates that this child is the default child, to be used when the attribute state matches no values in any children.

For instance, the child of the hatch switch bead that represents a closed hatch might be used when the hatch attribute value is either 0 (not-applicable), 1 (closed), or any value other than those enumerated in other children. This child would be commented:

```
@dis state 0,1,-1
```

The child with the popped hatch geometry should be used when the value is 2 (popped) or 3 (popped with person visible) and is commented:

```
@dis state 2,3
```

The child that contains the open hatch geometry should be used when the value is 4 (open) or 5 (open with person visible) and is commented:

```
@dis state 4,5
```

There may be many switch beads for the same attribute. For example, there may be several pieces of a tank's geometry that are affected by the destroyed attribute.

The WRM Model Specification also discusses the inheritance of appearance attributes from ancestor beads. The 3D Front End does not support this functionality. To tell the 3D Front End to, for example, display a camouflaged tank when the paint scheme attribute is set to camouflage, and a normal tank otherwise, the switch bead method described above must be used.

E.4.1. Appearance Attributes Supported by 3D Front End Models

The 3D Front End models support the following appearance attributes:

- ♦ All entity types:
 - mobility
 - fire_power
 - damage
 - trailing_effects
 - hatch
 - moving
 - paint_scheme (HLA only)
 - density (HLA only)
- ♦ Platform entities support:
 - frozen_status
 - power_plant_status
- ♦ Land platform entities:
 - launcher
 - camouflage_type
 - concealed
 - head lights, tail lights, brake lights, blackout lights, blackout brake lights, spot lights, interior lights
- ♦ Air platform entities:
 - afterburner
 - landing lights, navigation lights, anticollision lights, formation lights, spot lights, interior lights

- ♦ Munition entities:
 - launch_flash
- ♦ Non-lifeform entities:
 - smoke
 - flaming
- ♦ Surface
 - running lights, spot lights, interior lights
- ♦ Subsurface
 - running lights
- ♦ Sensor
 - lights, blackout lights, interior lights
- ♦ Cultural features
 - exterior lights, interior lights
- ♦ Lifeform entities:
 - life_form_state
 - weapon_1
 - weapon_2
 - flashlight
 - compliance
 - camouflage_type
 - health.

E.5. Attached Parts

An entity might have one or more attached parts – separate entities that are “attached” to one another. For instance, a missile might be attached to a launcher. As far as the 3D Front End is concerned, the “carrier” and “carried” entities are represented using separate models. The Model Specification includes sections on building models that can carry attached parts and models that can be carried as attached parts.

E.5.1. Building Models that Can Have Attached Parts

When building a model that can have attached parts, you must indicate where on the model the attachment stations are. At each attachment station, place a DOF bead. The DOF should be placed so that if the attached model were to be placed under it, it would have the correct position and orientation. Think of the DOF as a hanger from which to hang the attached part. If an attachment station is on an articulated part, for example, any model attached to the station should move along with the articulated part – the attachment station DOF should appear below the articulation DOF for that part.

The DOF must have one child, which is a group bead. This group must be commented with:

```
@dis attachment_station stationNum
```

where *stationNum* is the enumeration for the particular type of attachment station in question. You can use a value of -1 for *stationNum* to indicate that this is the default station. The group bead should have no children.

E.5.2. Building Models that Can Be Attached Parts

When building a model that can be an attached part, you must indicate the position of the attaching point, that is, the point on the model that should coincide with the attachment station point on the carrying model when it is attached. The position of the attachment point is indicated by placing a DOF bead there. The DOF should be at the very top of the model’s bead hierarchy, above any geometry, but geometrically positioned at the attaching point.

The attaching point’s DOF bead must have one child, which is a group bead. The group bead should be commented with:

```
@dis attachment_point
```

E.6. Animation

By default, model animation sequences are displayed as quickly as possible. However, you can configure the animation sequence rate using a WRM animation comment in the animation node's comment field.

The comment syntax is as follows:

```
@dis animation num_frames_per_second [ skip | noskip ]
```

where:

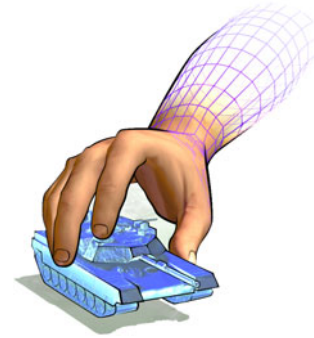
- ♦ `animation` tells the 3D Front End that animation instructions follow.
- ♦ `num_frames_per_second` is the rate at which the 3D Front End should display the frames.
- ♦ `skip` and `noskip` indicate whether the elapsed time should be used to choose the next frame, or whether at most one frame should be advanced each frame.

E.7. WRM Model Examples

The 3D Front End comes with many entity models that have articulated parts, multiple appearances, LOD nodes, and animations. Some good examples to look at are:

- ♦ `.data\models\makModels\M1A2_Abrams\M1A2_Abrams.flt` – This model contains a switch bead for damage state along with DOF beads for an articulated turret.
- ♦ `.data\models\makModels\Civilian_Male\Civilian_Male.flt` – This model contains a "walking" animation.

Most models have varying LOD levels.



Index

- disVersion command-line option 10-5
- mirror command-line option 10-7
- noDefaultMtl command-line option 11-3
- l command-line option 11-3, 12-6
- N command-line option 11-4

Numerics

- 1 (wildcard) 7-8
- 3D Front End
 - entity IDentity
 - ID 10-4
 - visibility to other applications 10-2

A

- absolute
 - model path 13-13
 - pathname 3-6
 - timestamp 11-13
- ACF file 10-11
 - editing 11-23
- Add Soil Type dialog box 6-12
- adding
 - category 5-22
 - component to object in OPD Editor 6-9
 - entities to Create menu B-5
 - entity to object parameter database 7-16
 - new preconfigured aggregate 7-25
 - object in OPD Editor 6-8
 - scenarios to Scenario Merge project 4-3
 - sensor signature 8-4
 - sensor type 8-4
 - soil type parameter 6-12

- subordinate object 6-11
 - symbol set 5-11
 - system to entity 5-16
- Additional Texture Paths column 13-3
- additional-opds parameter 3-5
- addMulticastAddr parameter 2-13, 2-17
- addSymbolMapFile parameter 2-5, 2-19
- advisory message
 - enabling and disabling 2-13, 2-15, 2-17
- agg-icon-map command 12-4
- aggregate A-8
 - adding new preconfigured type 7-25
 - adding preconfigured to object parameter
 - database 7-23
 - connection mode 2-23
 - dimension
 - setting threshold for entity 1-15
 - editing preconfigured 7-24
 - entity 12-4
 - formations 8-13
 - highest level
 - displaying 11-12
 - levels to display 11-5
 - maximum to display 11-10
 - model mapping 11-15, 12-4
 - parameter-type 7-23
 - route 11-6
 - time to display 11-5
 - volume
 - color 11-9
- aggregate dimensions 1-15
- aggregateDisplayLevel parameter 11-5
- aggregate-formation-controller parameter 8-13
- aggregateMap.mtl file 11-15, 11-21, 12-4

- aggregateModelRange** parameter 11-5
- aggregatesFilename** parameter 2-5
- aggregates.mtl* file 2-4
- aggregateTimeout** parameter 11-5
- all
 - connection mode 2-23
- alternative model set 12-6
- altitude
 - eyepoint 11-12
- ammo** parameter 8-12
- ammo select table 8-12
- angle
 - damage 8-9
- angle-of-incidence** parameter 8-9
- animated
 - icons 2-4
 - image
 - configuring symbols for 2-20
- animation E-9
 - configuring symbol map 2-20
 - model E-9
- appearance attribute E-9
 - list of supported E-6
 - specifying configuration file 2-5
- appearanceFile** parameter 2-5
- appearance.mtl* file 2-4
- appldRange** parameter 2-13, 2-17
- application
 - seeing eyepoint 10-2
- application number 2-5, 3-10
 - back-end 2-6
 - back-end for remapping objects 2-11
 - configuring 2-14
 - remapping for batch mode 2-12
- appNumber** parameter 2-5, 2-14, 11-5
- arbitrary offset UTM D-2
- arc
 - segment 11-6
- area
 - displaying 11-6
- argument
 - language 2-7
 - usingGlobalMiniMap 2-10
- articulated part 12-7, E-2, E-4, E-9
- askSaveScenarioIfChanged** parameter 2-5
- asynchronous I/O 2-14, 2-18
 - enabling or disabling 11-13
 - using for performance 1-13
- asynchronous terrain loading 11-10
- attached part 12-7, E-8

- attribute
 - appearance E-6
- autoRemapInCombinedMode** parameter 2-10
- autoRemapInSeparateMode** parameter 2-11
- auto-reorganize** parameter 3-5
- availableTaskSetRetrievalScheme** parameter 2-6
- avoiding
 - objects 8-20

B

- back-end
 - application number 2-6, 2-11
 - console 2-15
 - discovery by front-end 2-7
 - heartbeat 2-16
 - multiple 3-10
 - reloading OPD 2-15
 - remapping for batch mode 2-11
 - site ID 2-6
 - for remapping 2-11
 - specifying name of 2-6
 - timeout interval 2-6
- Back-end Settings tab 1-7
- backendApplicationName** parameter 2-6
- backendAppNum** parameter 2-6
- backendSiteId** parameter 2-6
- backendTimeoutInterval** parameter 2-6
- ballistic gun
 - controller descriptor 8-12
- batch file
 - configuring 2-14
- batch mode
 - remapping application number 2-12
 - remapping back-end 2-11
- batchFile** parameter 2-14
- batchModeModeRemapAppNum** parameter 2-12
- batchModeModeRemapSiteId** parameter 2-11
- batchScenarioFilename** parameter 2-6
- bead
 - DOF E-4, E-8
 - switch E-5
- biological cloud C-2
- bitmap
 - overlay logo 11-11
 - displaying 11-7
- blockAtNearClip** parameter 11-5
- blockedDistance** parameter 11-5
- blocking distance 11-5
- boundingVolume.mtl* file 2-4, 2-22

specifying path 2-6
boundingVolumesFile parameter 2-6
 broadcast
 enabling or disabling for Stealth 11-8
 broadcasting 10-2
 Broadcasting tab 10-2, 10-4

C

capabilities
 configuration file 2-6
capabilitiesFile parameter 2-6
capabilities.mtl file 2-4
catastrophic-kill parameter 8-9
 category
 adding 5-22
 changing 5-11
 changing echelon 7-26
 removing 5-22, 5-23
 renaming 5-22, 5-23
 undoing changes 5-23
category parameter 7-16, 7-26
 CDE-Motif look and feel 13-12
 cgf
 tuning tick() sleep interval 2-14
cgfDispatcherReceivePort parameter 2-14
 changing
 component priority 6-10
 Create Menu Tree 6-5
 default pathname 13-12
 echelon ID category and echelon ID level 7-26
 entity
 category 5-11
 enumeration 5-13
 force 5-12
 icon 5-8
 image 5-8
 name 5-13
 parameter 5-14
 symbols 2-21
 view in OPD Editor 6-4
 frame rate mode 3-5
 interface style 13-12
 path name 13-8
 site ID 3-10
 symbol set 5-9
 terrain database for scenario 3-7
 chemical cloud C-2
 civilian mapping UTM D-3
 class

DtSimComponentManager 7-9
DtSimObjectNetInterface 7-9
DtSimObjectStateRepository 7-9
DtSimTaskManager 7-9
DtVrfObject 7-6, 7-9
DtVrfObjectBaseStateRepository 7-6
 filtering
 FOM 11-9
 interaction
 filtering 11-9
 clipping plane
 near 11-11
 specifying 11-8
 clock mode 3-5
 closing
 scenario file in Scenario Merge 4-7
 cloud
 hazard C-2
 layer
 displaying 11-6
 clustering
 configuring 11-16
clusterRulesMap.mtl file 11-16
 coefficient probability 8-8
 collision avoidance
 configuring 8-20
 color
 aggregate volume 11-9
 detonation 11-5
 line 11-9
 friendly force label 11-9
 intervisibility lines 11-10
 overlay 11-11
 sensor volume 11-6
 text label 11-13
 trajectory histories 11-9
 combined mode
 remapping objects 2-10
 command
 agg-icon-map 12-4
 load-font 11-21
 model-map 11-15, 12-2
 trail-map 12-4
 command line
 running Scenario Merge from 4-8
 specifying parameters from 11-3
 command-line option 11-3
 --disVersion 10-5
 -l 11-3, 12-6
 --mirror 10-7

- N 11-4
- noDefaultMtl 11-3
- commandLineDiscoveryTimeout** parameter 2-7
- comment
 - MTL parameter 2-2
- comm-model-name** parameter 8-18
- commModelParams.mtl* file 2-23, 9-3, 9-4
- commonObjects.mtl* file 2-4, 2-21
- communication model 2-23, 8-18
 - custom 2-24
 - simple-radio-comm-model 2-23
- company name
 - specifying 11-11
- component 7-9
 - adding to object in OPD Editor 6-9
 - creating connection in OPD Editor 6-10
 - deleting in OPD Editor 6-7
 - descriptor
 - elements of 7-13
 - editing in OPD Editor 6-9
 - entity 7-12
 - priority 6-10
 - tuning 1-14
 - performance 1-14
- Component Manager A-5
- componentCodeGeneratorConfig.mtl* file 2-4
- component-descriptor-type 7-13
- component-manager** parameter A-5, A-7
- component-type 7-13
- conditional expression 3-8
 - entity type pattern 2-8
- configuration file 2-4
 - aggregateMap.mtl* 11-21
 - appearance attribute 2-5
 - boundingVolume.mtl* 2-4, 2-22
 - capabilities 2-6
 - clusterRuleSet.mtl* 11-16
 - commModelParams.mtl* 2-23, 9-3, 9-4
 - for object parameter database 5-4
 - loading from command line 11-3
 - ModelMap-billboards.mtl* 11-15
 - ModelMap.mtl* 12-2
 - overlayMenu.mtl* 2-22, B-2
 - overlayPalette.mtl* 2-22
 - params.mtl* 1-11, 11-5, 11-21
 - plugin.mtl* 2-21
 - rangeRings.mtl* 2-21
 - reading in Model Mapper 13-10
 - saving mappings to 13-9
 - settings 2-7
 - soundmap.mtl* 2-21
 - Stealth.mtl* 11-4, 11-21
 - symbolic names for 12-4
 - symbolmap-images.mtl* 2-4
 - tacticalGraphicsMap.mtl* 11-16
 - trailMap.mtl* 12-4
 - vrf.ent* 2-4, B-2, B-5
 - vrfSim.mtl* 1-15, 2-14
- configuring
 - aggregate symbol 12-4
 - back-end
 - application number 2-6
 - console 2-15
 - heartbeat 2-16
 - name 2-6
 - site ID 2-6
 - timeout interval 2-6
 - batch file 2-14
 - clustering 11-16
 - communications model 2-23
 - context sensitivity for tasks and sets 2-6
 - Create menu 2-22
 - default scenario path 2-7
 - destination address 2-13, 2-18
 - detonation and fire effects 1-11
 - DIS 2-13, 2-17, 2-18
 - dismounted infantry 11-16
 - drainInput 2-7, 2-16
 - DIS 2-13
 - HLA 2-13
 - drainInput for DIS 2-18
 - drainInput for HLA 2-17
 - emitter 8-16
 - entity labels 1-11
 - environmental data updates 2-15
 - ethernet address for file save 2-7
 - exercise ID 2-13, 2-18
 - exercise time 2-15
 - External Communications Model 9-4
 - feature avoidance 8-21
 - FED file name 2-12, 2-17
 - federation execution 2-12, 2-17
 - name 2-12, 2-17
 - FOM Mapper 2-12, 2-17
 - formations 8-13
 - frame rate buffer 2-12
 - frequency of data sends 2-15
 - hazard cloud 11-17
 - heartbeat threshold 2-13, 2-17
 - HLA 2-12, 2-17

- joysticks 8-19
- language 2-7
- location of help file 2-8
- logo 11-11
- map datum 2-10, 2-17
- MILSTD 2525B icons 12-6
- model loading 1-11
- multicast address 2-13, 2-17
- multi-channel display 10-11
- network frame length 2-12
- object avoidance 8-20
- obstruction avoidance 8-20
- overlay menu 2-7
- Overlay Palette 2-22
- path to OPD file 2-15
- performance 2-11
- port number 2-13, 2-18
- radio message model 2-23
- radios 8-18
- range rings 2-9
- range volumes 11-17
- receive port 2-14
- relative scenario paths 2-7
- save scenario prompt 2-5
- scenario file to use 2-16
- scenario save port 2-9
- sensors 8-2
- session ID 2-16
- settings 2-7
- simulation management message 2-16
- simulation time 2-16
- site ID 2-16
- sleep interval 2-16
- sound 1-11, 2-7, 2-21, 11-16
- spatial subdivision 1-17
- start mode 2-16
- symbols 2-12
- tactical graphics 11-16
- terrain database 11-16
 - path 2-15
 - to load 2-16
- tick management 2-12
- time stamp order 2-13, 2-17
- track history 2-11
- transmitter 2-8
- transmitters 8-18
- TrueType fonts for icons 11-20
- view control message 11-12
- VR-Forces for external communications effects
 - server 9-2
- confirmation prompt
 - Model Mapper 13-13
- connection
 - component 6-10
 - sensor 8-2
- connection mode 2-23
 - aggregate 2-23
 - all 2-23
 - force 2-23
- connection model
 - external 2-23
- consecutiveOverFrameRateThreshold** parameter 2-11
- consecutiveTimesOverThreshold** parameter 1-10
- console
 - configuring for back-end 2-15
 - enabling and disabling 11-7
 - enabling or disabling
 - in Windows 2-10
- context sensitivity
 - configuring 2-6
- control object 3-10, A-7
 - parameter-type 7-27
- controlling
 - remote 10-6
- Create Connection dialog box 6-10
- Create Entry dialog box 6-8
- Create menu B-2
 - adding entity to B-5
 - assigning or removing entity 5-14
 - configuring 2-22, B-2
 - editing 5-19, B-4
 - reorganizing B-4
 - reverting 5-22
- Create Menu Tree
 - changing 6-5
 - OPD Editor 6-4
- Create New Component dialog box 6-9
- Create Resource dialog box 6-11
- create-entity-menu-item B-5
- creating
 - component connection 6-10
 - custom model mapping file 12-6
 - entity B-5
 - from existing 5-18
 - new 5-18
 - model map 12-6
 - Scenario Merge project 4-3
 - simulation model set 5-6
 - user-defined formation 8-15
- cultural-feature-param** parameter A-8

- custom
 - communication model 2-24
- customizing
 - object parameter database 7-28
- cutltural-feature-param parameter 7-11
- cutural-feature-state-repository 7-11

D

- daemon
 - running vrfSim as 2-14
- daemon parameter 2-14
- damage
 - configuring entity to take 8-13
 - probability table 8-9
- damage actuator 8-9
- damage file 5-4, 7-27
- damage model 5-15, 5-16
 - specifying 5-17
- damage state DOF bead E-9
- data
 - configuring frequency of sends 2-15
 - environmental
 - updates 2-15
 - input 7-12
 - reading 11-6
- database D-2
 - object parameter 7-2
 - origin
 - specifying 11-18
 - part 11-19
- date
 - specifying 11-5
- date parameter 11-5
- datum
 - map 11-19
- day
 - time 11-13
- decayTrackHistory parameter 2-11
- default
 - pathname 13-8, 13-12
 - scenario 2-7
 - value
 - parameter 3-11
- default.acf* file 10-11
- defaultFreehandSamplingRate parameter 2-7
- defaultHeartbeatThreshold parameter 2-13, 2-17
- defaultMainLoopSleepInterval parameter 2-14
- defaultObjectTimeout parameter 2-13, 2-18
- defaultOverlayMenu parameter 2-7

- defaultParameterDatabasePath parameter 2-15, 3-6
- defaultScenario parameter 2-7
- defaultSettingsFile parameter 2-7
- default.sms* file 5-4, 5-6
- defaultSymbolMapFile parameter 2-7
- defaultTerrainDatabasePath parameter 2-15, 3-6
- defaultTimeThreshold parameter 11-5
- definition
 - parameter 6-5
- degrees of freedom E-4
- delaying
 - model loading 11-7
- deleting
 - category 5-22, 5-23
 - entity from simulation model set 5-19
 - mapping 13-9, 13-11
 - parameters and components in OPD Editor 6-7
 - unused paths 13-11
- description
 - parameter 6-5
- destAddrString parameter 2-13, 2-18, 11-5
- destination address 2-13, 2-18
- destroyFedExec parameter 2-12, 2-17, 11-5
- destroying
 - federation execution 2-12, 2-17, 11-5
- detonate effect
 - maximum 11-11
- detonate interaction
 - display radius 11-9
- detonate-sound parameter 11-16
- detonation
 - color 11-5
 - effects
 - configuring 1-11
 - ground clamping 11-10
 - line
 - color 11-9
 - displaying 11-7
- detResultAirBurstColor parameter 11-5
- detResultDefaultColor parameter 11-5
- detResultEntityImpactColor parameter 11-5
- detResultGroundImpactColor parameter 11-5
- dialog box
 - Add Soil Type 6-12
 - Create Connection 6-10
 - Create Entry 6-8
 - Create New Component 6-9
 - Create Resource 6-11
 - Display Options 1-4
 - Edit Entity Mapping 13-10

- Frame Rate Monitor 1-7
- Model Mapper Preferences 13-13
- Networking Options 1-16
- New Scenario 3-3
- Open Model Set 5-7
- Preferences 13-12, 13-13
- Terrain View Options 1-12
- DIS
 - configuring 2-13, 2-17, 2-18
 - encoding information in OpenFlight format E-2
 - entity ID 11-5, 11-12
 - enumerations in model mapping 12-4
 - heartbeat 11-5
 - threshold 2-13, 2-17
 - multicast ttl 11-11
 - protocol version 11-6
 - specifying version 10-5
 - WRM specification E-1
- DIS Enumeration Document 7-6
- disabling
 - advisory messages 2-13, 2-15, 2-17
 - asynchronous I/O 11-13
 - console window 2-10, 11-7
 - creation of an entity type 5-14
 - frame rate statistics 2-15
 - full screen mode 11-7
 - global Minimap 2-10
 - ground clamping 11-7
 - hidden track histories 1-11
 - HLA advisories 11-13
 - lighting effects 11-7
 - mirroring 10-7
 - orientation clamping 11-7
 - sound 11-8
 - soundRange parameter 11-12
 - status message 11-8
 - TrueType font icons 11-13
 - view control messages 2-10, 11-8
 - wireframe mode 11-8
- dismounted infantry
 - models 11-16
- display
 - frame rate 1-4
 - setting 1-4
 - viewing 1-5
 - mirroring options 10-8
- Display Options dialog box 1-4
- displaying
 - aggregate route 11-6
 - areas 11-6
 - bitmap logo 11-7
 - cloud layer 11-6
 - detonation line 11-7
 - emission 11-7
 - entity label 11-7
 - fog 11-7
 - highest level aggregate 11-12
 - obstacle 11-7
 - other MAK 3D GUIs 10-5
 - overlay 11-7
 - overlay object 11-7
 - phase line 11-7
 - Pivot mode graphic 11-7
 - polygons 1-12
 - route 11-7
 - scenario file in Scenario Merge 4-6
 - sky dome 11-8
 - tactical graphics 11-8
 - label 11-8
 - texture 11-8
 - track histories 11-8
 - trailing effects 11-8
 - transparency effects 11-8
 - waypoint 11-8
- disPort parameter 2-13, 2-18, 11-6
- distance
 - intervisibility 11-10
- disVersion parameter 11-6
- docking
 - Model Mapper toolbar 13-5
 - toolbar 13-5
- DOF bead E-4, E-8
- doNotUseConsole parameter 2-15
- doNotUseSettingsFile parameter 2-7
- drainInput
 - configuring 2-7, 2-16
 - configuring for DIS 2-13, 2-18
 - configuring for HLA 2-13, 2-17
- drainInputMaxTime parameter 11-6
- drawing mode 1-12
- DtNotifyLevel parameter 2-7, 2-15
- DtSimComponentManager class 7-9
- DtSimObjectNetInterface class 7-9
- DtSimObjectStateRepository class 7-9
- DtSimTaskManager class 7-9
- DtThresholder 1-15
- DtVrfObject class 7-6, 7-9
- DtVrfObjectBaseStateRepository class 7-6
- dynamic model loading 11-11

E

-e command-line option 1-17

echelon

 changing level 7-26

echelon ID

 category and echelon ID level 7-26

echelon-level parameter 7-23, 7-26

Edit Entity Mapping dialog box 13-10

editing

 ACF file 11-23

 components in OPD Editor 6-9

 Create menu 5-19, B-4

 embarkation ingress and egress points 8-22

 entity model 5-8

 object map file 3-10

 parameters 5-14, 6-5

 parameters for multiple objects 6-7

 plan file 3-7

 preconfigured aggregate 7-24

 scenario file 3-3

 in Scenario Merge 4-7

 sensor 8-3

 sensor signature 8-4

 system properties 5-17

 systems 6-13

egress point 8-22

embarkation

 editing ingress and egress points 8-22

 occupancy-director-controller 8-22

emission 11-6

 displaying 11-7

 frequency 11-6

emissionsColorIndicates parameter 11-6

emissionsFarColor parameter 11-6

emissionsHighFrequency parameter 11-6

emissionsHighPrFrequency parameter 11-6

emissionsLowFrequency parameter 11-6

emissionsLowPrFrequency parameter 11-6

emissionsNearColor parameter 11-6

emissionsRadiusExponent parameter 11-6

emissionsRadiusScaleFactor parameter 11-6

emissionsSegmentAngle parameter 11-6

emissionsShowLines parameter 11-6

emissionTimeout parameter 11-6

emitter

 configuring 8-16

enableAggregateRoutesDisplay parameter 11-6

enableAreasDisplay parameter 11-6

enableClouds parameter 11-6

enableConsole parameter 11-7

enableDelayedModelLoading parameter 1-11, 11-7

enableDetonationLines parameter 11-7

enableEmissionsDisplay parameter 11-7

enableEmitterHandling parameter 2-7

enableEntityLabels parameter 11-7

enableFog parameter 11-7

enableFullScreen parameter 11-7

enableGroundClamping parameter 11-7

enableGroundOrientationClamping parameter 11-7

enableHistoryTracking parameter 2-11

enableLifeformsAlwaysUpright parameter 11-7

enableLighting parameter 11-7

enableLogoOverlay parameter 2-7

enableMouseLook parameter 11-7

enableObstaclesDisplay parameter 11-7

enableOverlay parameter 11-7

enableOverlayLogo parameter 11-7

enableOverlayObjectsDisplay parameter 11-7

enablePhaseLinesDisplay parameter 11-7

enablePivotGraphic parameter 11-7

enablePlugins parameter 11-7

enableRemoteGraphics parameter 11-7

enableRoutesDisplay parameter 11-7

enableSkyDome parameter 11-8

enableSound parameter 11-8

enableSoundSystem parameter 2-7

enableStealthBroadcast parameter 10-2, 11-8

enableTacticalGraphicLabels parameter 11-8

enableTacticalGraphicsDisplay parameter 11-8

enableTexture parameter 11-8

enableTrailingEffects parameter 11-8

enableTrajectoryHistories parameter 11-8

enableTransparency parameter 11-8

enableViewControl parameter 10-6, 11-8

enableViewModeMirroring parameter 10-7, 11-8

enableViewModeRecording parameter 11-8

enableViewModeStatusMessages parameter 11-8

enableWaypointsDisplay parameter 11-8

enableWireframe parameter 11-8

enabling

 advisory messages 2-13, 2-15, 2-17

 asynchronous I/O 11-13

 console window 2-10, 11-7

 creation of an entity type 5-14

 External Communications Model 9-3

 frame rate statistics 2-15

 full screen mode 11-7

 global Minimap 2-10

 ground clamping 11-7

- HLA advisories 11-13
- lighting effects 11-7
- mirroring 10-7
- orientation clamping 11-7
- radio transmitter 9-3
- sound 11-8
- soundRange parameter 11-12
- status message 11-8
- TrueType font icons 11-13
- view control messages 2-10, 11-8
- wireframe mode 11-8
- entity
 - adding new to parameter database 7-16
 - appearance E-2
 - category
 - changing 5-11
 - changing
 - force 5-12
 - name 5-13
 - symbol set 5-9
 - undoing changes 5-15
 - clamping to ground 11-10
 - components 7-12
 - configuring
 - damage by munitions 8-13
 - emitter 8-16
 - obstruction avoidance 8-20
 - creating from existing 5-18
 - creating in Entity Editor 5-18
 - creation
 - enabling and disabling 5-14
 - deleting from simulation model set 5-19
 - editing 5-3
 - icon
 - changing 5-8
 - mapping 2-21
 - image
 - changing 5-8
 - label
 - configuring 1-11
 - maximum 11-10
 - model
 - editing 5-8
 - examples E-9
 - models 5-4
 - setting update thresholds 1-15
 - Stealth
 - marking text 11-13
 - systems 5-15
 - adding 5-16
 - tick rate 1-14
 - timeout 11-8
 - type 12-2
 - enumeration 10-5
 - in Model Mapper 13-7
 - mapping to model file 13-6
 - symbolic name 12-3
 - vrf.ent* file 2-4, B-2, B-5
- Entity Editor 5-3
 - starting 5-5
- entity enumeration
 - changing 5-13
- entity ID 11-12
 - DIS 11-5
 - site ID 11-12
- entity label
 - displaying 11-7
- Entity List
 - OPD Editor 6-4
- Entity Mapping column 13-3
- Entity Palette
 - assigning or removing entity 5-14
- Entity Resource Manager 8-12
- Entity Tree
 - OPD Editor 6-4
- entity type
 - changing 5-13
 - enumeration
 - for conditional expressions 2-8
- Entity Type column 13-3
- Entity Types tab 13-6
- entityEditorUi.mtl* file 2-4
- entityModelRange** parameter 11-8
- entity.mst* 6-5
- entity-sound** parameter 11-16
- entityTimeout** parameter 11-8
- entity-type 12-2
- entity-type** parameter 8-7, B-5
- entVrfDataTimeThreshold** parameter 2-15
- enumeration 7-6, 12-2, 13-6, 13-7
 - DIS 12-4
 - entity 10-5
 - changing 5-13
 - entity-type 12-2
 - object type A-9
- Enumeration and Bit Encoded Values for Use with
Protocols for DIS Applications 7-6
- environmental data
 - frequency of updates 2-15
- environmental process

- messages C-2
- envVrfDataTimeThreshold** parameter 2-15
- ethernet address
 - for file save 2-7
- execName** parameter 2-12, 2-17, 11-8
- exercise ID 2-13, 2-18
 - specifying 11-8
- exercise time
 - configuring 2-15
- exerciseld** parameter 2-13, 2-18, 11-8
- exerciseStartTime** parameter 2-15
- exercise-start-time** parameter 3-5
- explosion
 - radius for displaying 11-9
- external communication model 2-23
- external communications effects server
 - configuring VR-Forces for 9-2
 - QualNet 9-4
- External Communications Model 9-3
 - configuring 9-4
 - enabling 9-3
- eyepoint
 - default position 11-20
 - mouse control 11-7
 - orientation 11-12
 - position 11-12
 - radius for loading models 11-8
 - range for sound 11-12
 - rotation threshold for updates 11-13
 - starting altitude 11-12
 - step size 11-14
 - translation threshold for updates 11-13
 - turn rate 11-14

F

- far fog slider 11-9
- farClippingPlane** parameter 11-8
- farFogRange** parameter 11-9
- feature
 - avoiding 8-20
 - configuring avoidance of 8-21
- FED file
 - name 2-12, 2-17
 - specifying 11-9
- federate
 - name 2-12, 2-17
 - specifying 11-9
- federateName** parameter 2-12, 2-17, 11-9
- federation execution

- destroying 2-12, 2-17, 11-5
- name 2-12, 2-17
- federation execution name
 - specifying 11-8
- fedFileName** parameter 2-12, 2-17, 11-9
- field of view
 - specifying 11-13
- file
 - aggregateMap.mtl* 11-15, 12-4
 - aggregates.tml* 2-4
 - appearance.mtl* 2-4
 - capabilities.mtl* 2-4
 - clusterRulesMap.mtl* 11-16
 - commonObjects.mtl* 2-4, 2-21
 - componentCodeGeneratorConfig.mtl* 2-4
 - configuration 13-9
 - configuration and Model Mapper 13-10
 - configuration of symbolic names 12-4
 - default.acf* 10-11
 - default.sms* 5-4
 - entityEditorUi.mtl* 2-4
 - FLT 13-7
 - formation 8-14
 - guipreferences.mtl* 11-17
 - hazardCloudMap.mtl* 11-17
 - infantryMap.mtl* 11-16, 12-4
 - joyParam.dat* 8-19
 - loading configuration from command line 11-3
 - ModelMap.mtl* 11-15, 12-4
 - ModelMapper.bat* 13-2
 - Models.mtl* 12-4
 - MTL 2-2
 - object map 3-10
 - order of battle 3-10
 - overlayMenu.mtl* 2-4
 - params.mtl* 11-5
 - plan 3-7
 - plugin.mtl* 2-4
 - rangeRingsMap.mtl* 11-17
 - rangeRings.mtl* 2-4
 - saved views 11-15
 - scenario 3-3, 3-6
 - simulation model set 5-4
 - soundMap.mtl* 11-16
 - soundmap.mtl* 2-4
 - Stealth.mtl* 11-4
 - symbolmap-bmp.mtl* 2-4
 - symbolmap-ttf.mtl* 2-4
 - tacticalGraphicsMap.mtl* 11-16
 - trailMap.mtl* 11-16, 12-4

- vrfsent* 2-4, B-2, B-5
- vrfgui.mtl* 2-4, 2-5
- vrfguisettings.mtl* 2-4
- vrfsim.mtl* 1-15, 2-4, 2-14
- vrfsim.opd* 7-2
- filterFomClasses** parameter 1-12, 11-9
- filterFomInteractions** parameter 1-12, 11-9
- filtering
 - FOM classes 11-9
 - HLA traffic 1-12
 - interactions 11-9
- fire effect
 - configuring 1-11
 - maximum 11-11
- fire interaction
 - display radius 11-9
- fireDetonateVisualsRange** parameter 1-11, 11-9
- fire-sound** parameter 11-16
- fixed-frame 3-5
- Fixed-Frame Best-Effort 3-5
- Fixed-Frame Run-To-Complete 3-5
- fixed-frame-run-to-complete 3-5
- fixed-wing-entity-param** parameter A-8
- FLT file 11-19, 13-7
 - mapping
 - to models 12-2
- fog
 - displaying 11-7
- FOM
 - filtering classes 11-9
- FOM Mapper
 - configuring 2-12, 2-17
 - initialization data 11-9
 - initialization string 2-12, 2-17
 - library 11-9
- fomMapperInitData** parameter 2-12, 2-17
- fomMapperInitializationData** parameter 11-9
- fomMapperLib** parameter 2-12, 2-17, 11-9
- font
 - label size 11-10
 - mapping 2-20
 - TrueType 11-20
 - symbol map file 11-13
- force
 - connection mode 2-23
 - entity
 - changing 5-12
 - ID 10-3
 - image 5-10
 - number to display 11-11
 - forceFriendlyAggregateColor** parameter 11-9
 - forceFriendlyDetonationColor** parameter 11-9
 - forceFriendlyHistoryColor** parameter 11-9
 - forceFriendlyLabelColor** parameter 11-9
 - forceNeutralAggregateColor** parameter 11-9
 - forceNeutralDetonationColor** parameter 11-9
 - forceNeutralHistoryColor** parameter 11-9
 - forceNeutralLabelColor** parameter 11-9
 - forceOpposingAggregateColor** parameter 11-9
 - forceOpposingDetonationColor** parameter 11-9
 - forceOpposingHistoryColor** parameter 11-9
 - forceOpposingLabelColor** parameter 11-9
 - forceParameterDbReload** parameter 2-15
 - force-type** parameter B-4, B-5
- format
 - OpenFlight E-2
- formation
 - configuring 8-13
 - file 5-4, 8-14
 - user-defined 8-15
- frame
 - length of 3-5
 - sending mirror messages each 11-12
- frame rate 2-11
 - configuring 2-12
 - display
 - setting 1-4
 - mode
 - specifying 3-5
 - setting network 1-5
 - statistics
 - enabling and disabling 2-15
 - tuning update processing 1-8
 - viewing network and display 1-5
- Frame Rate Monitor dialog box 1-7
- frame-mode** parameter 3-5
- frame-time** parameter 3-5
- free form mapping 13-10
- Free Form tab 13-10
- free-floating toolbar 13-5
- freehand line
 - number of points 2-7
- frequency
 - back-end statistics 1-7
 - emission 11-6
 - mirror messages 11-12
- front-end
 - waiting for back-end 2-7
- full screen mode
 - enabling or disabling 11-7

G

- geocentric
 - origin 11-18
- geodeticMapDatum** parameter 11-10, 11-19
- global Minimap
 - enabling and disabling 2-10
- Global-Plan** parameter 3-4
- gradientFadeLabelColor** parameter 11-10
- graphics
 - performance impact 1-13
- ground clamping 11-10
 - detonations 11-10
 - enabling or disabling 11-7
 - range of interest 11-10
 - recalculating 11-10
- ground vehicle
 - configuring obstruction avoidance 8-20
- groundClampDetonationTerrainHits** parameter 11-10
- groundClampingMaximumExtent** parameter 11-10
- groundClampingMinimumExtent** parameter 11-10
- groundClampThreshold** parameter 11-10
- ground-vehicle-param** parameter 7-6, A-8
- GUI
 - specifying user-defined formation in 8-15
- guipreferences.mtl* file 11-17
- Gui-Terrain-Database** parameter 3-4, 3-7

H

- hazard cloud
 - configuring 11-17
 - configuring performance 1-13
- hazardCloudMap.mtl* file 11-17
- header file
 - objTypeEnum.h* A-9
- heartbeat
 - back-end
 - configuring 2-16
 - DIS 2-13, 2-17
 - relationship to thresholds 1-16
 - specifying 11-5
- heavy network traffic 1-10, 2-11
 - configuring 2-12
- help file
 - configuring location for 2-8
- hideOffScreenSymbols** parameter 2-12
- highest level
 - aggregate 11-12
- hit probability file 7-27

- hit probability table 8-7
- HLA
 - advisories
 - enabling or disabling 11-13
 - filtering traffic 1-12
 - interaction
 - timestamp order 11-12
 - parameters 2-12, 2-17
 - Stealth entity name 11-12
- HLA name
 - specifying 10-3
- hostAddrString** parameter 2-7
- human-param** parameter 7-11, A-8
- human-state-repository 7-11

I

- icon
 - animated 2-4
 - changing entity 5-8
 - changing symbol map file 2-21
 - for all forces 5-10
 - force 11-11
 - instead of model 12-6
 - specifying type of 11-13
 - toolbar 13-5
 - TrueType font 11-20
- ID
 - entity 10-4, 11-12
 - force 10-3
 - site 2-16
- ignore list 4-4
- ignoreAdvisories** parameter 2-15
- image
 - entity
 - changing 5-8
 - for all forces 5-10
- improving
 - performance 1-11, 1-12
- infantry models 11-16
- infantryMap.mtl* file 11-16, 12-4
- information message 2-15
- ingress point 8-22
- initialization data
 - FOM Mapper 11-9
- initialization string
 - FOM Mapper 2-12, 2-17
- input
 - of data 7-12
 - port and port group 7-13

- inset view
 - maximum to display 11-11
- interaction
 - filtering 11-9
 - timestamp order 11-12
- interface
 - look and feel 13-12
- intersection
 - intervisibility 1-13
- intervisibility
 - distance 11-10
 - intersection 1-13, 2-22
 - line
 - color 11-10
- intervisibilityLineColor** parameter 11-10
- intervisibilityRange** parameter 11-10
- IP address
 - specifying for unicast 11-5
- isectorUpdateThreshold** parameter 11-10

J

- joy-device-type** parameter 8-19
- joyParam.dat* file 8-19
- joy-rot-gain** parameter 8-19
- joy-rot-invert** parameter 8-19
- joy-rot-sensitivity** parameter 8-19
- joystick
 - configuring 8-19
- joy-trans-gain** parameter 8-19
- joy-trans-invert** parameter 8-19
- joy-trans-sensitivity** parameter 8-19

K

- keyboard
 - sending view control messages for 10-10

L

- label
 - color 11-9
 - configuring entity 1-11
 - font size 11-10
 - maximum to draw 11-10
 - style 11-10
- label** parameter B-5
- labelFontSize** parameter 11-10
- labelStyle** parameter 11-10

- Lambert
 - origin 11-19
- language
 - configuring 2-7
- language** argument 2-7
- lateral-clearance** parameter 8-20
- leader-promotion-id** parameter 8-14
- length
 - trajectory history 11-13
- level
 - aggregate 11-5
- levels-of-detail
 - switch range 11-10
- library
 - FOM Mapper 11-9
- lifeform
 - upright 11-7
- lighting effects
 - enabling or disabling 11-7
- line
 - freehand
 - points in 2-7
 - keeping settings 2-10
- lisp A-2
- listing
 - components in parameters database 7-15
- loadAsynchronous** parameter 11-10
- load-font command 11-21
- loading
 - databases from MTL file 11-18
 - models 1-11, 11-7
 - radius 11-8
 - plug-ins 11-7
 - scenario
 - timeout 2-9
 - Scenario Merge project 4-5
 - specifying dynamic or load at startup 11-11
 - symbol map file 2-21
 - terrain
 - asynchronous or synchronous 11-10
- loadPluginIfVersionMismatch** parameter 2-15
- local
 - object 7-9
 - network interface 7-12
- local-objects** parameter 7-9, 7-10, A-4
- LOD
 - node E-9
 - switch range 11-10
- lodScale** parameter 11-10
- logFrameRateStatistics** parameter 1-7, 2-15

- logging
 - view mode changes 11-8
- logo
 - bitmap 11-11
 - configuring 11-11
 - displaying 11-7
- look and feel 13-12
- lookahead** parameter 2-13
- Lynx Prime 11-23

M

- MÄK products
 - controlling eyepoint from 10-6
- MÄK Technologies LISP 2-2
- makePathsRelative** parameter 2-7
- map
 - datum 11-19
 - configuring 2-10
- map datum
 - configuring 2-17
 - specifying 11-10
- mapping
 - deleting 13-11
 - entity to no model 13-8
 - entity types to model files 13-6
 - free form 13-10
 - hazard clouds 11-17
 - model to remote eyepoint 10-5
 - models with Model Mapper 13-2
- object
 - back-end 2-11
- objects to back-end 2-11
- range volumes 11-17
- removing 13-9
- saving 13-9
- sounds 11-16
- tactical graphics 11-16
- trailing effects 11-16
- trailing effects to entities 12-4
- ttf and utf8 fonts 2-20

- marking text 10-3
 - Stealth entity 11-13
- master 10-7
- maxDrainInputReads** parameter 2-13, 2-18
- maxDrainInputTime** parameter 2-7, 2-16
- maximum
 - aggregate models 11-10
 - fire and detonate effects 11-11
 - inset views to display 11-11
 - labels to draw 11-10
 - models to load 11-10
 - trail segments 11-11
- maximumSignalTimeDifferent** parameter 2-8
- maxLoadedEntityModelDistance** parameter 1-11
- maxNumberOfAggregateModels** parameter 11-10
- maxNumberOfEntityLabels** parameter 1-11, 11-10
- maxNumberOfEntityModels** parameter 1-11, 11-10
- maxNumberOfFireDetonateEffects** parameter 11-11
- maxNumberOfInsetViews** parameter 11-11
- maxNumberOfTrailModels** parameter 11-11
- mcastTtl** parameter 2-13, 2-18, 11-11
- memory 1-11
- menu
 - adding to Create B-5
- merging
 - scenarios 4-2, 4-6
- message
 - environmentatl process C-2
 - frequency of 2-15
 - information and warning 2-15
- mirror
 - frequency 11-12
- mirroring 10-9
- notification for Scenario Merge 4-7
- notification level 11-11
- sending and receiving 8-18
- state update 10-2
- view control 10-6

- message-timeout** parameter 9-4
- Metal look and feel 13-12
- migrating
 - object parameter database to new release 7-28
- milgrid UTM D-3
- MILSTD 2525B icon 11-20
- minDrainInputTime** parameter 2-13, 2-17
- minimum tick period 1-14
- minimum tick period variance 1-14
- min-tick-period** parameter 1-14
- min-tick-period-variance** parameter 1-14
- minTrackHistoryMovement** parameter 2-11
- Mirror button 10-7
- mirroring 10-11
 - display options 10-8
 - each frame 11-12
 - enabling and disabling 10-7
 - messages supported 10-9
 - sending arbitrary message 10-7
 - sending messages every frame 10-10
 - sending option PDUs 11-12

- missile-param parameter A-8
 - mode
 - wireframe 11-8
 - model
 - aggregate
 - mapping 11-15
 - animation E-9
 - communication 2-23
 - custom mapping file 12-6
 - delaying loading 11-7
 - dismounted infantry 11-16
 - entity 5-4
 - editing 5-3, 5-8
 - examples E-9
 - infantry 11-16
 - loading 1-11, 11-11
 - mapping
 - to FLT files 12-2
 - to none 13-8
 - mapping to remote eyepoint 10-5
 - maximum aggregate 11-10
 - maximum to load 11-10
 - object 5-4
 - simulation 5-4
 - model file
 - mapping to entity type 13-6
 - Model File column 13-3
 - Model Mapper 12-2, 12-4, 13-2
 - overwrite prompt 13-13
 - starting 13-2
 - window 13-3
 - model mapping
 - creating 12-6
 - file
 - creating 12-6
 - symbolic name 12-3
 - model path
 - absolute or relative 13-13
 - Model Path column 13-3
 - model set 12-6
 - Model Specification for DIS Interoperability E-2
 - model-map command 11-15, 12-2
 - ModelMap-billboards.mtl* file 11-15
 - ModelMap.mtl* file 11-15, 12-2, 12-4
 - ModelMapper.bat* file 13-2
 - modelset-2525B.mtl* 12-6
 - modelset-2525B-ttf.mtl* 12-6
 - Models.mtl* file 12-4
 - ModSAF UTM D-3
 - monitoring
 - network and display frame rates 1-5
 - mouse
 - moving eyepoint 11-7
 - movement model 5-15
 - specifying 5-17
 - movement system 5-16
 - moving
 - model mapper toolbar 13-5
 - toolbar 13-5
 - moving object parameter A-8
 - moving-object-param** parameter 7-23, A-8
 - moving-object-param parameter type A-8
 - MST file 6-5
 - MS-Windows look and feel 13-12
 - MTL file 2-2
 - model mapping 12-2
 - vrfSim.mtl* 2-14
 - multicast address
 - configuring 2-13, 2-17
 - multicast time-to-live 2-13, 2-18
 - specifying 11-11
 - multi-channel 10-7
 - configuring 10-11
 - multipart databases 11-19
 - multiple
 - back-ends 3-10
 - munition
 - configuring entity for damage by 8-13
 - munition-type parameter 8-12
- ## N
- name
 - changing
 - entity 5-13
 - component 7-13
 - FED file 2-12, 2-17
 - federate 2-12, 2-17
 - Stealth in HLA 11-12
 - symbolic 12-3
 - near clipping plane
 - blocked distance 11-5
 - specifying 11-11
 - net-interface-min-tick-period** parameter 1-15, A-5, A-7
 - net-interface-min-tick-period-variance** parameter 1-15, A-5, A-7
 - netsocket
 - receive buffer size 11-11
 - send buffer size 11-12
 - network

- frame
 - configuring time to execute 2-12
- frame rate 1-4
 - setting 1-5
 - viewing 1-5
- processing time slider 1-8
- traffic
 - heavy 1-10
- visibility of Stealth on 11-8
- network card
 - multiple 2-7
- network interface A-5
 - hierarchy 7-11, 7-12
- networkFrameRateBuffer** parameter 1-10, 2-12
- Networking Options dialog box 1-16
- network-interface** parameter A-5, A-7
- networkTimeThreshold** parameter 1-10, 2-12
- New Scenario dialog box 3-3
- no model
 - mapping to 13-8
- non-offset UTM D-3
- notification level 2-15
 - Scenario Merge 4-7
 - specifying 11-11
- notifyLevel** parameter 11-11
- numTtfIconForces** parameter 11-11, 11-21

O

- object
 - adding
 - resources in OPD Editor 6-11
 - configuring
 - avoidance of 8-20
 - editing
 - multiple 6-7
 - mapping to back-end 2-10, 2-11
 - models 5-4
 - parameter A-3
 - default value 3-11
 - setting spatial subdivision 1-17
 - subcomponents 7-9
 - super-type 7-7
 - timeout threshold 2-13, 2-18
 - type 7-6
 - hierarchy 7-8
- object map file 3-10
- object parameter database 5-4, 7-2
 - adding
 - component in OPD Editor 6-9

- new entity 7-16
 - new object in OPD Editor 6-8
 - preconfigured aggregate 7-23, 7-25
 - soil type parameter 6-12
 - subordinate object 6-11
- configuring path to 2-15
- configuring sensors 8-2
- loading in OPD Editor 6-4
- migrating to new release 7-28
- pathname interpretation 7-27
- reloading 2-15
- saving 6-13

Object Parameter Database Editor. *See* OPD Editor

- object type enumeration A-9
- objectConsoleHistoryLimit** parameter 2-8
- objectInformationToolbar** parameter 2-8
- Object-Map** parameter 3-4
- object-type** parameter 7-8, 7-16, A-3
- objTypeEnum.h* header file A-9
- obstacle
 - configuring avoidance of 8-20
 - displaying 11-7
- obstruction
 - configuring avoidance of 8-20
- occupancy-director-controller
 - role in embarkation 8-22
- OPD
 - reloading 2-15
- OPD Editor 6-2
 - adding
 - new object 6-8
 - resources 6-11
 - adding subordinate object 6-11
 - changing
 - component priority 6-10
 - list views 6-4
 - compared to Entity Editor 5-3
 - connecting components 6-10
 - deleting parameters and components 6-7
 - editing
 - components 6-9
 - embarkation data 8-22
 - multiple objects 6-7
 - preconfigured aggregate 7-24
 - editing parameters 6-5
 - loading object parameter database 6-4
 - restoring values 6-7
 - saving object parameter database 6-13
 - starting 6-3

- OPE file 7-27
- Open Model Set
 - dialog box 5-7
- OpenFlight
 - file 13-7
 - format E-2
- opening
 - configuration file in Model Mapper 13-10
 - simulation model set 5-7
- optimal performance level 1-8
- optimizing
 - performance 1-11
- option
 - mirroring 10-8
- option PDU
 - sending when mirroring is enabled 11-12
- order of battle
 - file 3-10
 - default values in 3-11
- Order-Of-Battle** parameter 3-4
- orientation
 - eyepoint 11-12
- orientation clamping
 - enabling or disabling 11-7
- origin 11-18
 - flat earth 11-19
 - geocentric 11-18
 - Lambert 11-19
 - specifying 11-18
 - UTM 11-18
- outline
 - segment 11-6
- output
 - port group 7-13
- overlay
 - color 11-11
 - displaying 11-7
 - displaying logo 11-7
 - menu
 - default configuration file 2-7
 - name field 11-11
 - product name 11-11
 - specifying logo bitmap 11-11
- Overlay file 3-11
- overlay object
 - displaying 11-7
- Overlay Palette
 - configuring 2-22
- Overlay** parameter 3-4
- overlayColor** parameter 11-11
- overlayCompanyName** parameter 11-11
- overlayLogoFile** parameter 2-8, 11-11
- overlayLogoHeight** parameter 11-11
- overlayLogoPosition** parameter 2-8
- overlayLogoPositionX** parameter 2-8
- overlayLogoPositionY** parameter 2-8
- overlayLogoPreserveSize** parameter 11-11
- overlayLogoScaleFactor** parameter 2-8
- overlayLogoWidth** parameter 11-11
- overlayMenu.mtl* file 2-4, 2-22, B-2
 - specifying default 2-7
- overlayPalette.mtl* file 2-4, 2-22
- overlayProductName** parameter 11-11
- overwrite confirmation prompt 13-13
- ovl* file 3-11

P

- parameter
 - additional-opds 3-5
 - addMulticastAddr 2-13, 2-17
 - addSymbolMapFile 2-5, 2-19
 - aggregateDisplayLevel 11-5
 - aggregate-formation-controller 8-13
 - aggregateModelRange 11-5
 - aggregatesFilename 2-5
 - aggregateTimeout 11-5
 - ammo 8-12
 - angle-of-incidence 8-9
 - appearanceFile 2-5
 - appldRange 2-13, 2-17
 - appNumber 2-5, 2-14, 11-5
 - askSaveScenarioIfChanged 2-5
 - autoRemapInCombinedMode 2-10
 - autoRemapInSeparateMode 2-11
 - auto-reorganize 3-5
 - availableTaskSetRetrievalScheme 2-6
 - backendApplicationName 2-6
 - backendAppNum 2-6
 - backendSiteId 2-6
 - backendTimeoutInterval 2-6
 - batchFile 2-14
 - batchModeModeRemapAppNum 2-12
 - batchModeModeRemapSiteId 2-11
 - batchScenarioFilename 2-6
 - blockAtNearClip 11-5
 - blockedDistance 11-5
 - boundingVolumesFile 2-6
 - capabilitiesFile 2-6
 - catastrophic-kill 8-9

category 7-16, 7-26
cgfDispatcherReceivePort 2-14
commandLineDiscoveryTimeout 2-7
commenting out 2-2
comm-model-name 8-18
component list order 7-15
component-manager A-5, A-7
consecutiveOverFrameRateThreshold 2-11
consecutiveTimesOverThreshold 1-10
cultural-feature-param A-8
cutltural-feature-param 7-11
daemon 2-14
date 11-5
decayTrackHistory 2-11
defaultFreehandSamplingRate 2-7
defaultHeartbeatThreshold 2-13, 2-17
defaultMainLoopSleepInterval 2-14
defaultObjectTimeout 2-13, 2-18
defaultOverlayMenu 2-7
defaultParameterDatabasePath 2-15, 3-6
defaultScenario 2-7
defaultSettingsFile 2-7
defaultSymbolMapFile 2-7
defaultTerrainDatabasePath 2-15, 3-6
defaultTimeThreshold 11-5
definitions 6-5
deleting in OPD Editor 6-7
destAddrString 2-13, 2-18, 11-5
destroyFedExec 2-12, 2-17, 11-5
detonate-sound 11-16
detResultAirBurstColor 11-5
detResultDefaultColor 11-5
detResultEntityImpactColor 11-5
detResultGroundImpactColor 11-5
DIS 2-13, 2-17
disPort 2-13, 2-18, 11-6
disVersion 11-6
doNotUseConsole 2-15
doNotUseSettingsFile 2-7
drainInputMaxTime 11-6
DtNotifyLevel 2-7, 2-15
echelon-level 7-23, 7-26
editing in Entity Editor 5-14
editing in OPD Editor 6-5
emissionsColorIndicates 11-6
emissionsFarColor 11-6
emissionsHighFrequency 11-6
emissionsHighPrFrequency 11-6
emissionsLowFrequency 11-6
emissionsLowPrFrequency 11-6
emissionsNearColor 11-6
emissionsRadiusExponent 11-6
emissionsRadiusScaleFactor 11-6
emissionsSegmentAngle 11-6
emissionsShowLines 11-6
emissionTimeout 11-6
enableAggregateRoutesDisplay 11-6
enableAreasDisplay 11-6
enableClouds 11-6
enableConsole 11-7
enableDelayedModelLoading 1-11, 11-7
enableDetonationLines 11-7
enableEmissionsDisplay 11-7
enableEmitterHandling 2-7
enableEntityLabels 11-7
enableFog 11-7
enableFullScreen 11-7
enableGroundClamping 11-7
enableGroundOrientationClamping 11-7
enableHistoryTracking 2-11
enableLifeformsAlwaysUpright 11-7
enableLighting 11-7
enableLogoOverlay 2-7
enableMouseLook 11-7
enableObstaclesDisplay 11-7
enableOverlay 11-7
enableOverlayLogo 11-7
enableOverlayObjectsDisplay 11-7
enablePhaseLinesDisplay 11-7
enablePivotGraphic 11-7
enablePlugins 11-7
enableRemoteGraphics 11-7
enableRoutesDisplay 11-7
enableSkyDome 11-8
enableSound 11-8
enableSoundSystem 2-7
enableStealthBroadcast 10-2, 11-8
enableTacticalGraphicLabels 11-8
enableTacticalGraphicsDisplay 11-8
enableTexture 11-8
enableTrailingEffects 11-8
enableTrajectoryHistories 11-8
enableTransparency 11-8
enableViewControl 10-6, 11-8
enableViewModeMirroring 10-7, 11-8
enableViewModeRecording 11-8
enableViewModeStatusMessages 11-8
enableWaypointsDisplay 11-8
enableWireframe 11-8
entityModelRange 11-8

-
- entity-sound 11-16
 - entityTimeout 11-8
 - entity-type 8-7, B-5
 - entry A-2
 - entVrfDataTimeThreshold 2-15
 - envVrfDataTimeThreshold 2-15
 - execName 2-12, 2-17, 11-8
 - exerciseld 2-13, 2-18, 11-8
 - exerciseStartTime 2-15
 - exercise-start-time 3-5
 - farClippingPlane 11-8
 - farFogRange 11-9
 - federateName 2-12, 2-17, 11-9
 - fedFileName 2-12, 2-17, 11-9
 - filterFomClasses 1-12, 11-9
 - filterFomInteractions 1-12, 11-9
 - fireDetonateVisualsRange 1-11, 11-9
 - fire-sound 11-16
 - fixed-wing-entity-param A-8
 - fomMapperInitData 2-12, 2-17
 - fomMapperInitializationData 11-9
 - fomMapperLib 2-12, 2-17, 11-9
 - forceFriendlyAggregateColor 11-9
 - forceFriendlyDetonationColor 11-9
 - forceFriendlyHistoryColor 11-9
 - forceFriendlyLabelColor 11-9
 - forceNeutralAggregateColor 11-9
 - forceNeutralDetonationColor 11-9
 - forceNeutralHistoryColor 11-9
 - forceNeutralLabelColor 11-9
 - forceOpposingAggregateColor 11-9
 - forceOpposingDetonationColor 11-9
 - forceOpposingHistoryColor 11-9
 - forceOpposingLabelColor 11-9
 - forceParameterDbReload 2-15
 - force-type B-4, B-5
 - frame-mode 3-5
 - frame-time 3-5
 - geodeticMapDatum 11-10, 11-19
 - Global-Plan 3-4
 - gradientFadeLabelColor 11-10
 - groundClampDetonationTerrainHits 11-10
 - groundClampingMaximumExtent 11-10
 - groundClampingMinimumExtent 11-10
 - groundClampThreshold 11-10
 - ground-vehicle-param 7-6, A-8
 - Gui-Terrain-Database 3-4, 3-7
 - hideOffScreenSymbols 2-12
 - HLA 2-12, 2-17
 - hostAddrString 2-7
 - human-param 7-11, A-8
 - ignoreAdvisories 2-15
 - intervisibilityLineColor 11-10
 - intervisibilityRange 11-10
 - isectorUpdateThreshold 11-10
 - joy-device-type 8-19
 - joy-rot-gain 8-19
 - joy-rot-invert 8-19
 - joy-rot-sensitivity 8-19
 - joy-trans-gain 8-19
 - joy-trans-invert 8-19
 - joy-trans-sensitivity 8-19
 - label B-5
 - labelFontSize 11-10
 - labelStyle 11-10
 - lateral-clearance 8-20
 - leader-promotion-id 8-14
 - loadAsynchronous 11-10
 - loadPluginIfVersionMismatch 2-15
 - local-objects 7-9, 7-10, A-4
 - lodScale 11-10
 - logFrameRateStatistics 1-7, 2-15
 - lookahead 2-13
 - makePathsRelative 2-7
 - maxDrainInputReads 2-13, 2-18
 - maxDrainInputTime 2-7, 2-16
 - maximumSignalTimeDifferent 2-8
 - maxLoadedEntityModelDistance 1-11
 - maxNumberOfAggregateModels 11-10
 - maxNumberOfEntityLabels 1-11, 11-10
 - maxNumberOfEntityModels 1-11, 11-10
 - maxNumberOfFireDetonateEffects 11-11
 - maxNumberOfInsetViews 11-11
 - maxNumberOfTrailModels 11-11
 - mcastTtl 2-13, 2-18, 11-11
 - message-timeout 9-4
 - minDrainInputTime 2-13, 2-17
 - min-tick-period 1-14
 - min-tick-period-variance 1-14
 - minTrackHistoryMovement 2-11
 - missile-param A-8
 - moving-object-param 7-23, A-8
 - net-interface-min-tick-period 1-15, A-5, A-7
 - net-interface-min-tick-period-variance 1-15, A-5, A-7
 - networkFrameRateBuffer 1-10, 2-12
 - network-interface A-5, A-7
 - networkTimeThreshold 1-10, 2-12
 - notifyLevel 11-11
 - numTtfIconForces 11-11, 11-21

- object A-3
- objectConsoleHistoryLimit 2-8
- objectInformationToolBar 2-8
- Object-Map 3-4
- object-type 7-8, 7-16, A-3
- Order-Of-Battle 3-4
- Overlay 3-4
- overlayColor 11-11
- overlayCompanyName 11-11
- overlayLogoFile 2-8, 11-11
- overlayLogoHeight 11-11
- overlayLogoPosition 2-8
- overlayLogoPositionX 2-8
- overlayLogoPositionY 2-8
- overlayLogoPreserveSize 11-11
- overlayLogoScaleFactor 2-8
- overlayLogoWidth 11-11
- overlayProductName 11-11
- parameter-type 7-6, 7-9, 7-27, A-2, A-3
- params.mtl* file 11-5
- patternsFile 2-8
- performance 1-10
- Plan 3-4
- plan-manager A-5, A-7
- pluginsDirectory 2-8, 2-16
- position-offset 8-14
- preloadModels 11-11, 11-15
- preserveEmptyOverlayLayers 2-8
- processSignalInteractions 2-8
- promotion-id 8-14
- protocolVersion 10-5
- publishSimulationTimeForStealth 2-16
- publish-transmitter 8-18, 9-3
- qtAssistantLocation 2-8
- qtHelpFile 2-8
- random-number-seed 3-5
- range-determinant 8-7, 8-9
- range-list 8-7
- rangeRingsFile 2-9
- recvBufferSize 2-14, 2-18, 11-11
- remoteControllerReceivePort 2-9
- remote-objects 7-10, A-6
- respondToMyViewControlOnly 10-6, 11-12
- rotary-wing-entity-param A-8
- rprFomVersion 2-13, 2-17, 11-12
- runInTimeConstrainedMode 2-13
- runInTimeManagementMode 2-17
- runInTimeRegulatingMode 2-13
- saveTimeoutInterval 2-9
- scenario
 - file 3-4
 - scenarioFileName 2-9, 2-16
 - scenarioTimeoutInterval 2-9
 - SelectionGroups 3-4
 - sendBufferSize 2-14, 2-18, 11-12
 - sendFedTime 2-13, 2-17, 11-12
 - sendMirrorMessagesEveryFrame 10-10, 11-12
 - sendMirrorMessagesForOptions 11-12
 - separateModeRemapAppNum 2-11
 - separateModeRemapSiteld 2-11
 - separateObjectLists 2-9
 - sessionId 2-9, 2-16
 - set-data-request-type 3-9
 - ShowGroundTruthForForce 2-9
 - showSplashScreen 2-9
 - ShowSpotReportsForForce 2-9
 - simMgmtPduProcessLevel 2-16
 - simTime 2-16
 - simTimeScale 2-13
 - simulationModelSet 2-9, 2-16, 5-6
 - Simulation-Model-Set-Files 3-5
 - siteld 2-9, 2-16, 11-12
 - sleepInterval 2-16
 - slowNetworkFrameOverride 2-12
 - soundRange
 - enabling or disabling 11-12
 - soundRange 1-11, 11-12, 11-16
 - soundRangeEnabled 11-12
 - specifying at startup 11-3
 - SpotReportSystemEnabled 2-9
 - startAltitude 11-12, 11-20
 - startInRunMode 2-16
 - startOrientation 11-12, 11-20
 - startPosition 11-12, 11-20
 - state-repository A-4, A-6
 - state-repository-min-tick-period 1-15
 - state-repository-min-tick-period-variance 1-15
 - statusUpdateHeartbeat 2-16
 - stealthEntityIdentifier 10-4, 11-12
 - stealthEntityType 10-5, 11-12
 - stealthForcelIdentifier 10-3, 11-12
 - stealthHLAName 10-3, 11-12
 - stealthMarking 10-3, 11-13
 - stealthRotationThreshold 10-5, 11-13
 - stealthTranslationThreshold 10-5, 11-13
 - stopping-distance-factor 8-20
 - storeLineValuesMode 2-10
 - structure of entries A-2
 - substitute-message-size 9-4
 - surface-entity-param A-8

- system-id 8-17
- tacticalGraphicTimeout 11-13
- task-manager A-5, A-7
- task-type 3-9
- Terrain-Database 3-4
- terrainDatabase 2-10
- terrainFileName 2-16
- textLabelColor 11-13
- tickManagingEnabled 1-10, 2-12
- time-multiplier 3-4
- timeMultiplierLowerBound 2-10
- timeMultiplierUpperBound 2-10
- timeOfDay 11-13
- trackHistoryTimeout 2-11
- trailsBasedOnAppearance 11-13
- trajectoryHistoryLength 11-13
- ttfSymbolMap 11-13, 11-21
- updateHiddenTrajectories 1-11
- useAbsoluteTimeStamps 2-13, 2-14, 2-17, 2-18, 11-13
- use-actual-message-size 9-4
- useAdvisories 2-13, 2-17, 11-13
- useAsyncIO 1-13, 2-14, 2-18, 11-13
- useCustomMapDatum 2-10, 2-17
- useTtfIcons 11-13, 11-21
- usingConsole 2-10
- value
 - restoring in OPD Editor 6-7
- vector-obstruction-type 8-21
- vector-type 8-21
- verticalFieldOfView 11-13
- viewConstraint 11-14
- viewControlMessages 2-10
- viewDefaultOffset 11-14
- viewDefaultPitchOffset 11-14
- viewFloor 11-14
- viewStepSize 11-14
- viewTurnRate 11-14
- viewWideGroupMaxPsiRate 11-14
- vrf-base-object-param 7-27, A-3
- warhead 8-12
- whoToSmooth 11-14
- xrAggregateDisplayLevel 11-5
- xrDate 11-5
- xrEmissionsColorIndicates 11-6
- xrEmissionsRadiusExponent 11-6
- xrEmissionsRadiusScaleFactor 11-6
- xrEmissionsSegmentAngle 11-6
- xrEmissionsShowLines 11-6
- xrEnableCloudLayer 11-6
- xrEnableFog 11-7
- xrEnableGroundClamping 11-7
- xrEnableGroundOrientation Clamping 11-7
- xrEnableLighting 11-7
- xrEnableSkyDome 11-8
- xrEnableTexture 11-8
- xrEnableTrailingEffects 11-8
- xrEnableTrajectoryHistories 11-8
- xrEnableTransparency 11-8
- xrEnableWireframe 11-8
- xrFarClippingPlane 11-8
- xrFarFogRange 11-9
- xrGroundClampThreshold 11-10
- xrIntervisibilityRange 11-10
- xrLodScale 11-10
- xrNearClippingPlane 11-11
- xrShowHighestAggregatesOnly 11-12
- xrSmoothPeriod 11-12
- xrTimeOfDay 11-13
- xrVerticalFieldOfView 11-13
- xrViewStepSize 11-14
- xrViewTurnRate 11-14
- xrWhoToSmooth 11-14
- parameters database
 - component list order 7-15
- parameter-type
 - aggregate 7-23
 - control object 7-27
- parameter-type parameter 7-6, 7-9, 7-27, A-2, A-3
- params.mtl* file 1-11, 11-5, 11-21
- part 11-19
 - articulated and attached 12-7
- path
 - deleting unused 13-11
 - model 13-13
 - name
 - changing 13-8
 - relative and absolute 2-7
- Path Name column 13-3
- pathname
 - default 13-8, 13-12
 - interpreting by object parameter database 7-27
 - scenario files 3-6
- pattern
 - for conditional expressions 2-8
- patternsFile parameter 2-8
- pause mode 2-16
- PDU
 - sending 10-2
 - siman

- configuring 2-16
- view control 10-6
- performance 1-12
 - asynchronous I/O 1-13
 - configuring 2-11
 - factors affecting 1-3
 - hazard cloud images 1-13
 - heavy network traffic level 1-10
 - intervisibility intersections 1-13
 - level
 - setting 1-8
 - optimal 1-8
 - optimizing 1-3, 1-11
 - parameters 1-10
 - sleep time 1-17
 - spatial subdivision 1-17
 - tuning
 - cgf 2-14
 - network interface 1-15
 - state repository 1-15
- phase line
 - displaying 11-7
- pitch
 - eyepoint 11-14
- Pivot mode
 - displaying graphic 11-7
 - graphic 11-11
- plan
 - file
 - editing 3-7
- plan manager A-5
- Plan** parameter 3-4
- Plan View Display
 - controlling eyepoint from 10-6
- plan-manager** parameter A-5, A-7
- plug-in
 - loading 11-7
- plugin.mtl* file 2-4, 2-21
- pluginsDirectory** parameter 2-8, 2-16
- point
 - number in freehand line 2-7
- point-to-point communications
 - configuring 2-13, 2-18
- polygon
 - displaying 1-12
- port 7-13
 - for save scenario request 2-9
 - for scenario files 2-14
 - group 7-13
 - number 2-13

- port number 2-18
- position
 - eyepoint 11-12
 - eyepoint default 11-20
- position-offset** parameter 8-14
- precedence
 - components in parameters database 7-15
- preferences
 - saving 13-13
 - setting 13-11
- Preferences dialog box 13-12, 13-13
 - Model Mapper 13-13
- preloadModels** parameter 11-11, 11-15
- preserveEmptyOverlayLayers** parameter 2-8
- priority
 - component
 - changing 6-10
- probability-coefficient list 8-7
- processSignalInteractions** parameter 2-8
- product name
 - overlay 11-11
- project
 - adding scenarios to 4-3
 - creating Scenario Merge 4-3
 - file 4-5
 - Scenario Merge 4-3
 - loading Scenario Merge 4-5
 - removing scenarios from 4-3
 - saving Scenario Merge 4-5
 - specifying ignore list for 4-4
- promotion-id** parameter 8-14
- prompt
 - saving scenario 2-5
- properties
 - editing system 5-17
- protocol
 - DIS version 11-6
 - specifying for outgoing PDUs 10-5
- protocolVersion** parameter 10-5
- publishing
 - simulation time 2-16
- publishSimulationTimeForStealth** parameter 2-16
- publish-transmitter** parameter 8-18, 9-3

Q

- qtAssistantLocation** parameter 2-8
- qtHelpFile** parameter 2-8
- QualNet
 - using with VR-Forces 9-4

R

- radio
 - configuring 8-18
 - message
 - configuring 2-23
 - transmitter 8-18
- radio transmitter
 - enabling 9-3
- radius
 - displaying explosions 11-9
 - for loading aggregate models 11-5
 - model loading 11-8
- radius exponent
 - sensor volume 11-6
- radius scale
 - sensor volume 11-6
- random-number-seed** parameter 3-5
- range
 - sounds, for 11-12
 - Time Multiplier toolbar 2-10
- range ring
 - configuring 2-9
- range volumes
 - configuring 11-17
- range-determinant** parameter 8-7, 8-9
- range-list** parameter 8-7
- rangeRingsFile** parameter 2-9
- rangeRingsMap.mtl* file 11-17
- rangeRings.mtl* configuration file 2-4, 2-21
- reading
 - configuration file in Model Mapper 13-10
 - data 11-6
- recalculating
 - ground clamping 11-10
- receive buffer size 2-14, 2-18
 - specifying 11-11
- receiving
 - radio messages 8-18
- recording
 - view mode changes 11-8
- recvBufferSize** parameter 2-14, 2-18, 11-11
- reference ellipsoid 2-10
 - configuring 2-17
- refresh rate 1-4
- relative model path 13-13
- relative path 3-6
- relative timestamp 11-13
- remapping object
 - combined mode 2-10
 - separate mode 2-11
- remote
 - object 7-9, A-6
 - network interface 7-12
- remote control
 - by MÄK products 10-6
 - of the eyepoint 10-6
- remote graphics
 - configuring 11-7
- remoteControllerReceivePort** parameter 2-9
- remote-objects** parameter 7-10, A-6
- removing
 - category 5-22, 5-23
 - mapping 13-9
 - scenarios from Scenario Merge project 4-3
 - system 5-17
- renaming
 - category 5-22, 5-23
- reorganizing
 - Create menu B-4
- repository A-4
- reset position
 - distance from entity 11-14
 - pitch 11-14
- resource
 - adding to object 6-11
- respondToMyViewControlOnly** parameter 10-6, 11-12
- restoring
 - parameter values in OPD Editor 6-7
- reverting
 - category changes 5-23
 - Create menu 5-22
- rotary-wing-entity-param** parameter A-8
- rotation
 - setting threshold for entity 1-15
 - threshold for eyepoint updates 11-13
- rotation threshold 1-15
- route
 - aggregate 11-6
 - displaying 11-7
- RPR FOM 12-2
 - specifying 11-12
 - specifying version 2-13, 2-17
- rprFomVersion** parameter 2-13, 2-17, 11-12
- run mode 2-16
- runInTimeConstrainedMode** parameter 2-13
- runInTimeManagementMode** parameter 2-17
- runInTimeRegulatingMode** parameter 2-13

S

- sample size
 - back-end statistics 1-7
- save scenario prompt 2-5
- saved views file 11-15
- saveTimeoutInterval** parameter 2-9
- saving
 - mappings 13-9
 - object parameter database 6-13
 - preferences 13-13
 - Scenario Merge project 4-5
 - window layout 11-17
- saving scenario
 - timeout 2-9
- scenario
 - adding to Scenario Merge project 4-3
 - closing in Scenario Merge 4-7
 - default 2-7
 - displaying scenario file in Scenario Merge 4-6
 - editing 4-7
 - file
 - editing 3-3
 - parameters 3-4
 - pathnames 3-6
 - relative path 2-7
 - specifying which to use 2-16
 - loading
 - from command line 2-7
 - timeout 2-9
 - merging 4-2, 4-6
 - parameter
 - version** 3-5
 - removing from Scenario Merge project 4-3
 - saving 2-9
 - timeout 2-9
 - temporary directory 3-12
 - terrain database 3-7
- Scenario Merge 4-2
 - closing scenario file 4-7
 - command line syntax 4-8
 - console output 4-7
 - creating project 4-3
 - displaying scenario file 4-6
 - editing scenario file 4-7
 - ignore list 4-4
 - loading project 4-5
 - project file 4-3, 4-5
 - saving project 4-5
- scenarioFileName** parameter 2-9, 2-16
- scenarioTimeoutInterval** parameter 2-9
- segment
 - arc 11-6
 - outline 11-6
- Selection Groups file 3-11
- SelectionGroups** parameter 3-4
- send buffer size 2-14, 2-18
 - specifying 11-12
- sendBufferSize** parameter 2-14, 2-18, 11-12
- sendFedTime** parameter 2-13, 2-17, 11-12
- sending
 - data
 - configuring 2-15
 - radio messages 8-18
 - view control message 11-8
- sendMirrorMessagesEveryFrame** parameter 10-10, 11-12
- sendMirrorMessagesForOptions** parameter 11-12
- sensor 8-2
 - adding new 8-4
 - configuring 8-2
 - connecting 8-2
 - signature 8-4
 - signature-object-sensor 8-2
 - system definition file
 - editing 8-3
- sensor signature 5-16
- sensor system 5-15
- sensor volume
 - color 11-6
 - radius exponent 11-6
 - radius scale 11-6
- separate mode
 - remapping objects 2-11
- separateModeRemapAppNum** parameter 2-11
- separateModeRemapSitedId** parameter 2-11
- separateObjectLists** parameter 2-9
- session ID 2-9
 - configuring 2-16
- sessionId** parameter 2-9, 2-16
- set data request
 - configuring context sensitivity 2-6
- set statement 3-8
- Set to All 5-10
- set-data-request-type** parameter 3-9
- setting
 - back-end statistics frequency and sample size 1-7
 - display frame rate 1-4
 - network frame rate 1-5

- performance level 1-8
- preferences 13-11
- range of Time Multiplier toolbar 2-10
- settings
 - configuring 2-7
 - default 2-7
 - line creation 2-10
- sgr* file 3-11
- ShowGroundTruthForForce** parameter 2-9
- showSplashScreen** parameter 2-9
- ShowSpotReportsForForce** parameter 2-9
- signature
 - sensor 8-4
- signature-object-sensor 8-2
- simMgmtPduProcessLevel** parameter 2-16
- simple-radio-comm-model 2-23
- simTime** parameter 2-16
- simTimeScale** parameter 2-13
- simulation
 - start mode 2-16
- simulation management message
 - configuring 2-16
- simulation model set 5-4
 - adding symbol map file 5-11
 - creating 5-6
 - deleting entity 5-19
 - file 5-4
 - opening 5-7
- simulation time
 - configuring 2-16
 - publishing 2-16
- simulationModelSet** parameter 2-9, 2-16, 5-6
- Simulation-Model-Set-Files** parameter 3-5
- site ID 3-10
 - back-end 2-6
 - configuring 2-16
 - remapping back-end 2-11
 - remapping batch mode 2-11
 - specifying 2-9, 11-12
- siteld** parameter 2-9, 2-16, 11-12
- size
 - font for labels 11-10
- sky dome
 - displaying 11-8
- slave 10-7
- sleep interval 2-14
 - configuring 2-16
- sleep time
 - tuning 1-17
- sleepInterval** parameter 2-16
- slider
 - far fog 11-9
- slowNetworkFrameOverride** parameter 2-12
- smooth period 11-12
- smoothing
 - objects supported 11-14
- soil type
 - adding to object 6-12
- sound
 - configuring 2-7
 - range 1-11
 - enabling or disabling 11-8
 - mapping 11-16
 - range in which to play 11-12
- soundMap.mtl* file 11-16
- soundmap.mtl* file 2-4, 2-21
- soundRange parameter
 - enabling or disabling 11-12
- soundRange** parameter 1-11, 11-12, 11-16
- soundRangeEnabled** parameter 11-12
- spatial subdivision
 - configuring 1-17
- specifying 11-11
 - appearance attribute configuration file 2-5
 - boundingVolume.mtl* path 2-6
 - clipping plane 11-8
 - company name 11-11
 - database origin 11-18
 - date 11-5
 - default *default.sms* 5-6
 - exercise ID 11-8
 - FED file 11-9
 - federate name 11-9
 - federation execution name 11-8
 - field of view 11-13
 - frame length 3-5
 - frame rate mode 3-5
 - HLA name 10-3
 - map datum 11-10
 - movement model 5-17
 - multicast time-to-live 11-11
 - near clipping plane 11-11
 - notification level 11-11
 - parameters with startup command 11-3
 - Pivot mode graphic 11-11
 - receive buffer size 11-11
 - RPR FOM 11-12
 - send buffer size 11-12
 - site ID 2-9, 11-12
 - switch range 11-10

- type of icon 11-13
- UDP port 11-6
- unicast IP address 11-5
- update threshold 10-5
- user-defined formation in GUI 8-15
- view constraint 11-14
- view floor 11-14
- specifying default 5-6
- SpotReportSystemEnabled** parameter 2-9
- standard offset UTM D-3
- startAltitude** parameter 11-12, 11-20
- starting
 - Entity Editor 5-5
 - Model Mapper 13-2
 - OPD Editor 6-3
 - Scenario Merge tool 4-2
- startInRunMode** parameter 2-16
- startOrientation** parameter 11-12, 11-20
- startPosition** parameter 11-12, 11-20
- state repository A-4
 - hierarchy 7-10, 7-11
- state updates, sending 10-2
- statement
 - conditional 3-8
 - set 3-8
- state-repository** parameter A-4, A-6
- state-repository-min-tick-period** parameter 1-15
- state-repository-min-tick-period-variance** parameter 1-15
- status message
 - enabling or disabling 11-8
- statusUpdateHeartbeat** parameter 2-16
- Stealth
 - simulation clock 2-16
- Stealth Control Toolkit 10-6
- stealthEntityIdentifier** parameter 10-4, 11-12
- stealthEntityType** parameter 10-5, 11-12
- stealthForceIdentifier** parameter 10-3, 11-12
- stealthHLAName** parameter 10-3, 11-12
- stealthMarking** parameter 10-3, 11-13
- Stealth.mtl* file 11-4, 11-21
- stealthRotationThreshold** parameter 10-5, 11-13
- stealthSoundMap.mtl* 1-11
- stealthTranslationThreshold** parameter 10-5, 11-13
- step size
 - eyepoint 11-14
- stopping-distance-factor** parameter 8-20
- storeLineValuesMode** parameter 2-10
- style
 - interface 13-12
- subcomponent A-4
 - classes 7-9
- subordinate object
 - adding 6-11
- substitute-message-size** parameter 9-4
- super-type 7-7
- surface
 - damage specification 8-9
- surface-entity-param** parameter A-8
- switch bead E-5, E-9
- switch range
 - specifying 11-10
- symbol
 - animated 2-20
 - configuring 2-12
 - configuring aggregate 12-4
 - for all forces 5-10
- symbol map file 5-4
 - adding to simulation model set 5-11
 - animations 2-20
 - default 2-7
 - for fonts 2-20
 - loading 2-21
 - TrueType font 11-13
- symbol set
 - adding 5-11
 - changing 5-9
- symbolic name 12-2, 12-3, 13-12
- symbolmap-bmp.mtl* file 2-4
- symbolmap-images.mtl* file 2-4
- symbolmap-ttf.mtl* file 2-4
- synchronizing
 - VR-Forces and Stealth simulation time 2-16
- synchronous
 - terrain loading 11-10
- system
 - adding to entity 5-16
 - editing 6-13
 - editing properties 5-17
 - entity 5-15
 - removing 5-17
 - sensor 8-2
- system definition file
 - sensor 8-3
- system-id** parameter 8-17
- Systems List
 - OPD Editor 6-4

T

- tab
 - Entity Types 13-6
 - Free Form 13-10
- tactical graphics
 - displaying 11-8
 - label
 - displaying 11-8
 - mapping 11-16
 - timeout 11-13
- tacticalGraphicsMap.mtl* file 11-16
- tacticalGraphicTimeout** parameter 11-13
- target-type parameter 8-12
- task
 - configuring context sensitivity 2-6
- task-manager** parameter A-5, A-7
- task-type** parameter 3-9
- temporary directory
 - scenario 3-12
- terrain database D-2
 - asynchronous or synchronous loading 11-10
 - changing for a scenario 3-7
 - configuration files 11-16
 - configuring 2-16
 - configuring path to 2-15
 - loading from MTL file 11-18
- Terrain View Options dialog box 1-12
- Terrain-Database** parameter 3-4
- terrainDatabase** parameter 2-10
- terrainFileName** parameter 2-16
- text label
 - color 11-13
- textLabelColor** parameter 11-13
- texture 13-3
 - displaying 11-8
- threshold
 - data updates 2-15
 - DIS heartbeat 2-13, 2-17
 - entity update 1-15
 - object timeout 2-13, 2-18
 - update 10-5
- tick
 - configuring 2-12
 - rate
 - tuning 1-14
 - sleeping between 2-16
- tickManagingEnabled** parameter 1-10, 2-12
- time
 - day 11-13
 - multiplier
 - range 2-10
 - simulation
 - setting 2-16
- Time Multiplier toolbar 2-10, 3-4
- time stamp order
 - configuring 2-13, 2-17
- time-multiplier** parameter 3-4
- timeMultiplierLowerBound** parameter 2-10
- timeMultiplierUpperBound** parameter 2-10
- timeOfDay** parameter 11-13
- timeout
 - entity 11-8
 - for sending save message 2-9
 - interval
 - back-end 2-6
 - object 2-13, 2-18
 - scenario load 2-9
 - tactical graphics 11-13
- timestamp
 - absolute or relative 11-13
 - order
 - HLA interaction 11-12
- toolbar 13-5
 - docking 13-5
 - moving model mapper 13-5
 - Time Multiplier 2-10
- track history
 - configuring 2-11
 - disabling 1-11
 - displaying 11-8
- trackHistoryTimeout** parameter 2-11
- trail segment
 - maximum to display 11-11
- trailing effects
 - displaying 11-8
 - drawn per entity 11-13
 - mapping 11-16
 - mapping to entities 12-4
- trail-map command 12-4
- trailMap.mtl* file 11-16, 12-4
- trailsBasedOnAppearance** parameter 11-13
- trajectory history
 - color 11-9
 - length 11-13
 - updating hidden 11-13
- trajectory history. See track history
- trajectory smoothing
 - specifying smooth period 11-12
- trajectoryHistoryLength** parameter 11-13

- translation
 - setting threshold for entity 1-15
 - threshold 1-15
 - threshold for eyepoint updates 11-13
- transmitter
 - configuring 2-8, 8-18
 - radio 8-18
- transparency effects
 - displaying 11-8
- TrueType font 11-20
 - enabling use of 11-13
 - mapping 2-20
 - symbol map file 11-13
- ttfSymbolMap** parameter 11-13, 11-21
- tuning
 - components 1-14
 - performance
 - component 1-14
 - network interface 1-15
 - state repository 1-15
 - sleep time 1-17
 - update processing 1-8
- turn rate
 - eyepoint 11-14

U

- UDP port
 - specifying 11-6
- undoing
 - category changes 5-23
 - changes to Create menu 5-22
 - entity edits in Entity Editor 5-15
- unicast
 - specifying IP address 11-5
- unicode font
 - mapping 2-20
- unit
 - icon
 - TrueType font 11-20
- Universal Transverse Mercator D-2
- update
 - eyepoint rotation 11-13
 - eyepoint translation 11-13
 - frequency of 2-15
 - messages 10-2
 - processing 1-8
 - threshold
 - specifying 10-5
- updateHiddenTrajectories** parameter 1-11

- updating
 - trajectory histories 11-13
- upright
 - lifeform 11-7
- useAbsoluteTimeStamps** parameter 2-13, 2-14, 2-17, 2-18, 11-13
- useActualMessageSize** parameter 9-4
- useAdvisories** parameter 2-13, 2-17, 11-13
- useAsyncIO** parameter 1-13, 2-14, 2-18, 11-13
- useCustomMapDatum** parameter 2-10, 2-17
- USE-DEFAULT keyword 3-11
- user-defined
 - formation 8-15
- useTtfIcons** parameter 11-13, 11-21
- USGS UTM D-3
- usingConsole** parameter 2-10
- usingGlobalMiniMap** argument 2-10
- UTM D-2
 - arbitrary offset D-2
 - milgrid D-3
 - MODSAF D-3
 - non-offset D-3
 - origin 11-18
 - standard offset D-3
 - USGS or civilian mapping D-3

V

- vapor cloud C-2
- variable
 - tick rate 1-14
- variable-frame 3-5
- Variable-Frame Run-To-Complete 3-5
- vector
 - feature
 - configuring avoidance of 8-21
- vector-obstruction-type** parameter 8-21
- vector-type** parameter 8-21
- version
 - DIS 11-6
 - DIS of 10-5
 - RPR FOM 2-13, 2-17
- version** parameter
 - scenario file 3-5
- verticalFieldOfView** parameter 11-13
- view
 - saved 11-15
- view constraint
 - specifying 11-14
- view control message 10-6, 10-7

- configuring 11-12
- enabling and disabling 2-10
- enabling or disabling 11-8
- enabling processing of message 10-6
- mirroring messages supported 10-9
- options 10-8
- script 10-6
- sending 11-8
- sending every frame 10-10
- sending explicitly 10-7
- view floor
 - specifying 11-14
- view mode
 - changes
 - recording 11-8
 - reset position
 - distance 11-14
 - pitch 11-14
- viewConstraint** parameter 11-14
- viewControllerMessages** parameter 2-10
- viewDefaultOffset** parameter 11-14
- viewDefaultPitchOffset** parameter 11-14
- viewFloor** parameter 11-14
- viewing
 - network and display frame rates 1-5
 - other MAK 3D GUIs 10-5
- viewStepSize** parameter 11-14
- viewTurnRate** parameter 11-14
- viewWideGroupMaxPsiRate** parameter 11-14
- visibility
 - of eyepoint to other applications 10-2
- vrf B-2
- vrf-base-object-param** parameter 7-27, A-3
- vrf-base-object-param parameter type A-7
- vrf.ent* file 2-4, B-2, B-5
 - editing B-4
 - reorganizing B-4
- vrfGui.mtl* file 2-4, 2-5
- vrfGuiSettings.mtl* file 2-4
- vrf-object-param parameter type A-7
- VR-Forces
 - controlling eyepoint from 10-6
- vrfSim
 - running as daemon 2-14
- vrfSim.mtl* file 1-15, 2-4, 2-14, 5-6
- vrfSim.opd* file 7-2

W

- warhead** parameter 8-12

- warning message 2-15
- waypoint
 - displaying 11-8
- weapon system 5-15, 5-16
- WGS84 11-10, 11-19
- whoToSmooth** parameter 11-14
- Wide Group mode
 - heading change rate 11-14
- wildcard 7-8, 12-2
- window
 - layout
 - saving 11-17
 - Model Mapper 13-3
- wireframe mode
 - enabling or disabling 11-8
- WRM Entity Model Specification 12-7, E-1, E-6

X-Y-Z

- XR mode
 - configuring clustering 11-16
- xrAggregateDisplayLevel** parameter 11-5
- xrDate** parameter 11-5
- xrEmissionsColorIndicates** parameter 11-6
- xrEmissionsRadiusExponent** parameter 11-6
- xrEmissionsRadiusScaleFactor** parameter 11-6
- xrEmissionsSegmentAngle** parameter 11-6
- xrEmissionsShowLines** parameter 11-6
- xrEnableCloudLayer** parameter 11-6
- xrEnableFog** parameter 11-7
- xrEnableGroundClamping** parameter 11-7
- xrEnableGroundOrientationClamping** parameter 11-7
- xrEnableLighting** parameter 11-7
- xrEnableSkyDome** parameter 11-8
- xrEnableTexture** parameter 11-8
- xrEnableTrailingEffects** parameter 11-8
- xrEnableTrajectoryHistories** parameter 11-8
- xrEnableTransparency** parameter 11-8
- xrEnableWireframe** parameter 11-8
- xrFarClippingPlane** parameter 11-8
- xrFarFogRange** parameter 11-9
- xrGroundClampThreshold** parameter 11-10
- xrIntervisibilityRange** parameter 11-10
- xrLodScale** parameter 11-10
- xrNearClippingPlane** parameter 11-11
- xrShowHighestAggregatesOnly** parameter 11-12
- xrSmoothPeriod** parameter 11-12
- xrTimeOfDay** parameter 11-13
- xrVerticalFieldOfView** parameter 11-13
- xrViewStepSize** parameter 11-14

Index

`xrViewTurnRate` parameter 11-14

`xrWhoToSmooth` parameter 11-14



Link - Simulate - Visualize

VRF-3.11-7-080104