B-HAVE Module

for VR-Forces

Users Guide

VT MÄK
A company of VT Systems

B-HAVE Module for VR-Forces

Users Guide

VT MÄK
A company of VT Systems

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

VT MÄK
150 Cambridge Park Drive, 3rd Floor
Cambridge, MA 02140 USA

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

info@mak.com

www.mak.com

Revision BHV-4.3-1-150217

Contents

Preface

Getting Started

B-HAVE Module Concepts

Generating PathData

Using the B-HAVE Module in VR-Forces

Generating Traffic

B-HAVE Messages API

Index

Preface

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

Please see release documentation for information about changes and updates to the B-HAVE Module since this manual went to press.

How This Manual is Organized

The chapters are organized as follows:

Chapter 1, *Getting Started* introduces the features of the B-HAVE Module and explains how to install it and set up license management.

Chapter 2, *B-HAVE Module Concepts* explains the how the B-HAVE Module is able to improve path planning and collision avoidance and analyze topology.

Chapter 3, *Generating PathData* explains how to prepare terrain databases for use with the B-HAVE Module.

Chapter 4, *Using the B-HAVE Module in VR-Forces* explains how to use the tasks and set data requests that the B-HAVE Module adds to VR-Forces as well as other B-HAVE functions you can use during a simulation.

Chapter 5, *Generating Traffic* describes how to create background traffic using the spawn and sink feature (also called Pattern of Life).

Chapter 6, *B-HAVE Messages API* describes extensions to the VR-Forces remote control API that you can use to manage the B-HAVE Module.

B-HAVE Module Documentation

Documentation is provided in PDF format. Manuals are in the *./doc* directory. The B-HAVE Module documentation set is as follows:

- *B-HAVE Module for VR-Forces Users Guide* describes how to install and use the B-HAVE Module program.
- *B-HAVE Module for VR-Forces Release Notes*. Release documentation consists of release notes.

MÄK Products

The B-HAVE Module for VR-Forces is a member of the VT MÄK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MÄK 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) and the Distributed Interactive Simulation (DIS) protocol. 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 (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you a 2D and 3D views of a simulated environment.

  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.

- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes three end-user applications (VR-Vantage Stealth, VR-Vantage PVD, and VR-Vantage IG) and the VR-Vantage Toolkit.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to entities so that it moves as they do.
  - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.

  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.

- 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 GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
  - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
  - DI-Guy Scenario™. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.
  - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
  - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.
  - DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.

How to Contact Us

For B-HAVE Module 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
VR-Vantage support:	vrv-support@mak.com

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses.html
Product version and platform information:	www.mak.com/support/product-versions.html
For the free, unlicensed MÄK RTI:	www.mak.com/resources/bonus-material/cat_view/16-bonus-materials/24-mak-high-performance-rti.html
MÄK Community Forum:	www.mak.com/community-forum/1-forum.html

Post

Send postal correspondence to:	VT MÄK 150 Cambridge Park Drive, 3rd Floor Cambridge, MA, USA 02140

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.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
`{ }`	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 Linux 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.
Italic	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**.
	Click the icon to run a tutorial video in the default browser.
	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 B-HAVE Module home directory. For example, the directory *vrforces4.3/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

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

Third Party Licenses

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 that is distributed under the Boost License. All header files that contain Boost code are properly attributed. The Boost web site 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 that 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. For more information about these libraries please see the following web sites:

- 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/

Kynapse License

Drainage, Visual Landscape, Visual Survey, Visual Toolbox, Visual LISP, Voice Reality, Volo, Vtour, Wire, Wiretap, WiretapCentral, XSI, and XSI (design/logo).

The following are registered trademarks or trademarks of Autodesk Canada Co. in the USA and/or Canada and other countries: Backburner, Multi-Master Editing, River, and Sparks.

The following are registered trademarks or trademarks of Moldflow Corp. in the USA and/or other countries: Moldflow MPA, MPA (design/logo), Moldflow Plastics Advisers, MPI, MPI (design/logo), Moldflow Plastics Insight, MPX, MPX (design/logo), Moldflow Plastics Xpert.

All other brand names, product names or trademarks belong to their respective holders.

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS". AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OR MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

1. Getting Started

This chapter provides a brief introduction to the B-HAVE Module and explains how to install it.

1.1. Introduction

The B-HAVE Module for VR-Forces[1] is a VR-Forces plug-in that extends the movement models for lifeforms and ground platforms. It improves:

- 3D path finding.
- Simplicity of use of basic agents (GoTo).
- Complexity and realism of behaviors (Hide, Flee, Wander).
- Complexity and realism of scenarios.

The B-HAVE Module is powered by Kynapse® from Autodesk®. It is based on Kynapse's fundamental concepts of Topological Modeling for Path Finding, and Spatial Reasoning for Advanced Behaviors, which are integrated into VR-Forces as advanced planning capabilities.

The B-HAVE Module offers VR-Forces users the ability to:

- Make entities move autonomously without specifying routes.
- Make entities navigate through complex environments, indoors and outdoors, while avoiding collisions with obstacles and other entities.
- Quickly populate a 3D environment with dozens of entities wandering in buildings, on roads, and on pavement (pattern of life).
- Develop complex scenarios in which entities hide from each other and flee from threats.

1.1.1. The B-HAVE Module Architecture

The B-HAVE Module is implemented as a plug-in for VR-Forces. The front-end plug-in allows you to generate PathData to associate with terrain databases. Entities use the PathData to understand the environment and plot a course. It also allows you to use the B-HAVE Module's tasks and run scripts through the GUI. The back-end plug-in performs all the AI-related runtime computations.

1. B-HAVE stands for Brains for Human Activity in Virtual Environments.

1.1.2. Using the B-HAVE Module in a Simulation

Before you can use the B-HAVE Module in a simulation, you must generate PathData for a terrain database. Then, you create a scenario using the updated terrain.

- The B-HAVE Module includes PathData for several of the terrains shipped with VR-Forces, including Makland, VR-Village, and the Hawaii inset in VR-TheWorld Online.
- The B-HAVE Module does not support aggregate-level simulation.

You create a scenario the same way that you would any VR-Forces scenario, except that lifeforms and ground platforms can now use B-HAVE Module tasks for movement in addition to or instead of the standard VR-Forces tasks.

1.2. Installing the B-HAVE Module

The B-HAVE Module is provided as an executable installation program for Windows and a compressed tar file for Linux. Please note the following important installation criteria:

- You must install the B-HAVE Module into the directory in which you have installed VR-Forces. On Windows, the installer defaults to the default installation directory for the associated version of VR-Forces, for example, *C:\MAK\vrforces4.3*. If you did not install VR-Forces into the default directory, it is your responsibility to change the B-HAVE Module installation directory to your VR-Forces installation directory.

  If the installer warns you that you are installing into an existing directory, ignore the warning.

- Each version of the B-HAVE Module is built for a specific version of VR-Forces and will work only with that version. For example, B-HAVE Module 4.3 for VR-Forces 4.3 will work only with VR-Forces 4.3. It would not work with VR-Forces 4.2.

➤ To install the B-HAVE Module on Windows, run the installation executable and follow the directions of the installation wizard.

➤ To install the B-HAVE Module on Linux, uncompress the installation file and untar it into the directory in which you installed VR-Forces.

Table 1-1 lists the files and directories installed in the VR-Forces directory by the B-HAVE Module.

Table 1-1: Files and directories installed into VR-Forces by the B-HAVE Module

File/Directory	Description
./appData/settings/BHAVE	B-HAVE configuration file.
./userData/scenarios/B-HAVE_Demos	Sample scenarios that use B-HAVE Module tasks.
./userData/terrains/	PathData for terrain databases.
./bin	B-HAVE Module DLLs are installed in the *bin* directory.
./plugins/B-HAVE	Plug-in DLLs for the B-HAVE Module.

1.2.1. License Management for the B-HAVE Module

Like all MÄK products, the B-HAVE Module requires a license. For details about obtaining a license, please see Chapter 2, Installing VR-Forces, in *VR-Forces Users Guide*.

1.2.2. Localizing the B-HAVE Module

You can localize the B-HAVE Module menus using Qt Linguist, as described in Section 2.5, "Localizing the Graphical User Interface," in *VR-Forces Users Guide*. An untranslated file for B-HAVE is included in the installation.

2. B-HAVE Module Concepts

This chapter provides conceptual background for the B-HAVE Module. Understanding these concepts will help you to prepare terrains for use with the B-HAVE Module. Concepts will also help you understand the behaviors to be expected from entities executing B-HAVE Module tasks.

2.1. Topological Modeling

In VR-Forces, when an entity is given a movement task, such as Move to Location or Move to Waypoint, it ordinarily moves directly towards the destination, taking relatively little account of topography, vector data, or other entities, except for basic collision avoidance. If path planning is enabled, the entity moves along roads if they are present. If you want an entity to move along an optimal route, you must draw the route and task the entity to follow it.

When entities carry out B-HAVE Module tasks, they use previously computed topological models of the virtual 3D environment to compute, in real time, an optimized path to the destination (path finding). As they follow the path, they automatically avoid obstacles and other entities. The topological models are called PathData.

2.2. How PathData is Generated

PathData is generated by the B-HAVE Module from within the VR-Forces front-end. Before you can use newly generated PathData, you must close your scenario and reopen it. (This is so it can be loaded by the back-end.) If the terrain is modified, the PathData must be generated again.

The B-HAVE Module analyzes the terrain and creates a navigation mesh (NavMesh) of the terrain. The NavMesh defines areas in which an entity can move, given its dimensions. A lifeform can move in spaces that a ground vehicle cannot move in. Figure 2-1 shows a NavMesh for a terrain that has several areas in which entities cannot move.

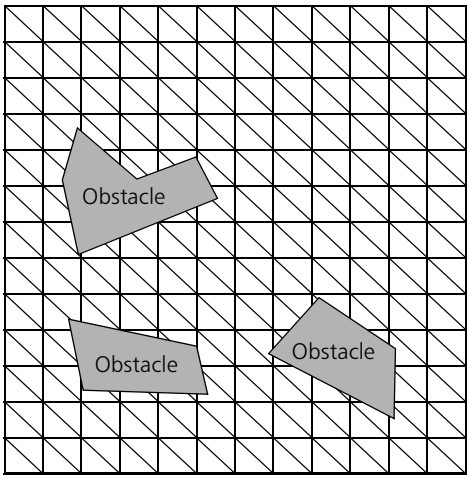


Figure 2-1. NavMesh

Based on the NavMesh, the B-HAVE Module generates a graph of locations in the terrain that the entity type can move to. Each location can be accessed from at least one other location. Figure 2-2 shows a hypothetical graph of the paths that an entity can follow in the NavMesh shown in Figure 2-1.

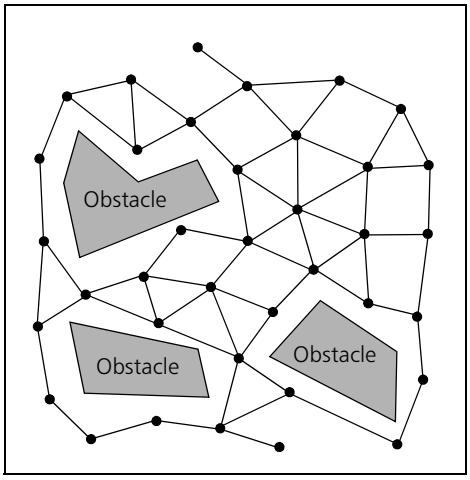


Figure 2-2. Graph

For details about generating path data, please see Chapter 3, *Generating PathData*.

2.3. Path Finding

Path finding is the process by which an entity plans a path, follows the path, and avoids obstacles along the path.

2.3.1. Planning a Path

Every time an entity decides to move (either through an autonomous decision or a task assignment), it computes its path. Path computation, or path planning consists of running an A* algorithm, with a given heuristic, on the PathData.

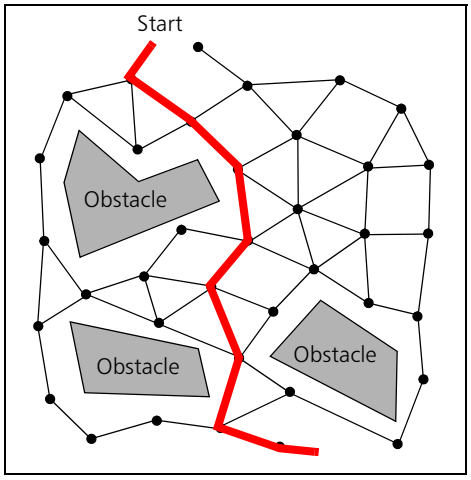


Figure 2-3. Planning a path

A path finding heuristic is a cost estimate for the shortest path to reach a target. This cost can be measured according to various criteria, such as distance, time, and danger. The path finding heuristic is critical for the performance of the A* algorithm used by the B-HAVE Module.

The standard path finding heuristic is the Euclidian Heuristic. In this heuristic, the cost of moving from one vertex to another vertex is the Euclidian length of the connecting edge. The B-HAVE Module also uses a Road Constrained Heuristic, in which a ground platform computing its path is forced to move only on roads.

2.3.2. Path Following

Once a path is determined for an entity, it can start moving to its final goal. As illustrated in Figure 2-3, a path is a set of vertices and edges. However, entities do not necessarily follow the exact path. The B-HAVE Module smooths the entity's movement.

The first intermediary destination (or subgoal) of the entity is the nearest vertex. Before moving towards this point, the entity checks to see if it can move directly to the next vertex in its path. This test is performed against the AI mesh. If the next vertex is reachable, it becomes the new subgoal. The entity continues testing successive vertices until it finds a vertex that it cannot reach directly. The final trajectory is a smooth line that connects the various subgoals.

Figures 2-4 and 2-5 illustrate this process. Using the prospective path defined in Figure 2-3, the entity computes its actual path. In Figure 2-4, view 1, it tests the first three vertices (dashed lines) and finds that it can reach all of them directly. In Figure 2-4, view 2, it tests the fourth vertex and finds that it can reach it. It then tests the fifth vertex and finds it cannot reach it directly. Therefore, the fourth vertex becomes the subgoal for this portion of the path. Figure 2-5 shows the final calculated path (dashed line). Notice how the path to the fourth vertex is much smoother than a path that strictly follows the vertices and edges would be, because the entity determined that it could move directly to that vertex.

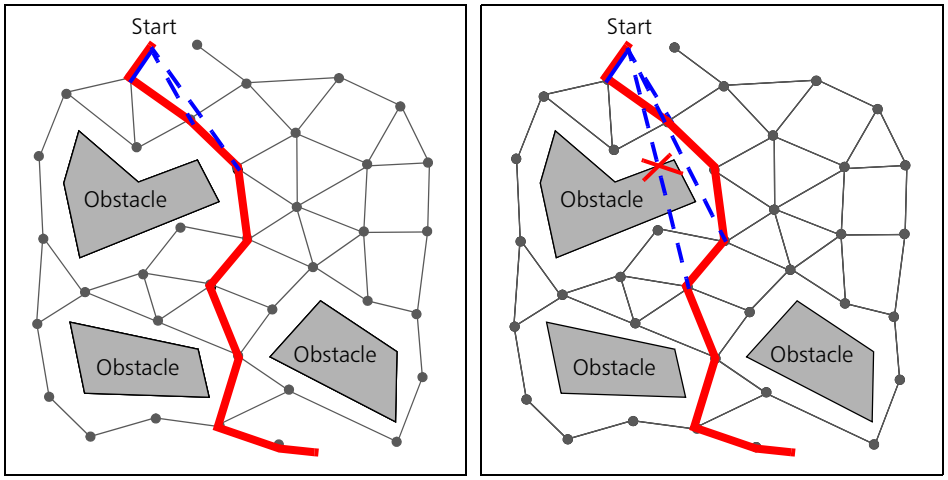


Figure 2-4. Checking next vertex

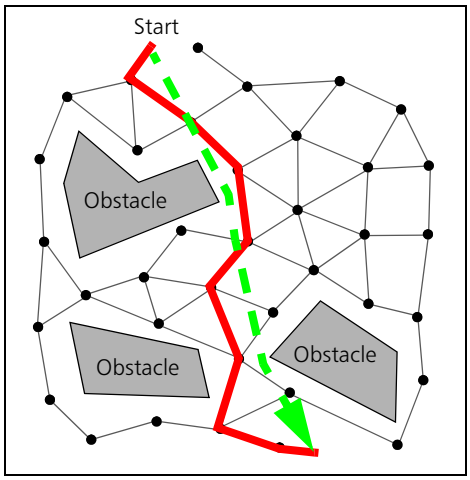


Figure 2-5. Final calculated path

2.3.3. Dynamic Avoidance

While following its path, an entity may encounter other entities. VR-Forces models include collision avoidance. However in complex scenarios, the scenario developer may have to spend a considerable amount of time creating routes and coordinating entities to minimize collisions among entities and obstacles.

The B-HAVE Module uses a dynamic avoidance algorithm to make entities behave realistically when they are about to collide. Entities detect potential collisions in advance and adjust their trajectories smoothly using the PathData.

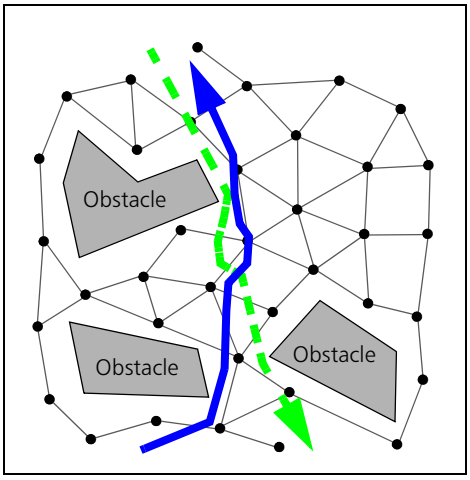


Figure 2-6. Collision avoidance

In Figure 2-6, two entities traveling in opposite directions avoid each other, then resume their routes. Compare the track of the dashed line to that in Figure 2-5.

2.4. Spatial Reasoning

Advanced entity behaviors require a certain level of understanding of the topology from each entity's point of view. This understanding, called spatial reasoning, allows an entity to decide where to go, to identify a hiding position, or to identify a threat point.

The B-HAVE Module can provide an analysis of the PathData in the vicinity of any point in the environment to find a collection of vertices or edges with properties that are relevant for decision purposes, for example:

- The nearest vertices that are accessible by the entity and are out of sight of point A. These are the nearest positions where the entity can hide from someone standing on A. These are also the nearest destinations if an entity wants to flee from point A.
- Edges connecting one point out of sight of the entity and another point within sight of the entity (but not necessarily accessible). These are edges that a sniper could move to, shoot at the entity, and then hide from it.

This dynamic topological analysis (DTA) can be used at any moment by any behavior, and may require frequent updates (every time that a movement of an entity changes its perception point of view significantly).

2.4.1. Dynamic Topological Analysis

DTA in the B-HAVE Module runs a search-for-the-best-candidate algorithm on the existing path data. DTA is optimized for low CPU cost and memory footprint. DTA starts from any given position and propagates to the nearest vertices following the connecting edges, testing edges or vertices for specific properties.

In Figure 2-7, a bot (any entity controlled by the B-HAVE Module) wants to hide from the Player (view 1). The analysis explores points in the vicinity of the bot, testing to see if they are in line-of-sight of the player (views 2 and 3), until it finds two hiding position candidates (view 4).

These tests are performed over several frames, to minimize their effect on performance.

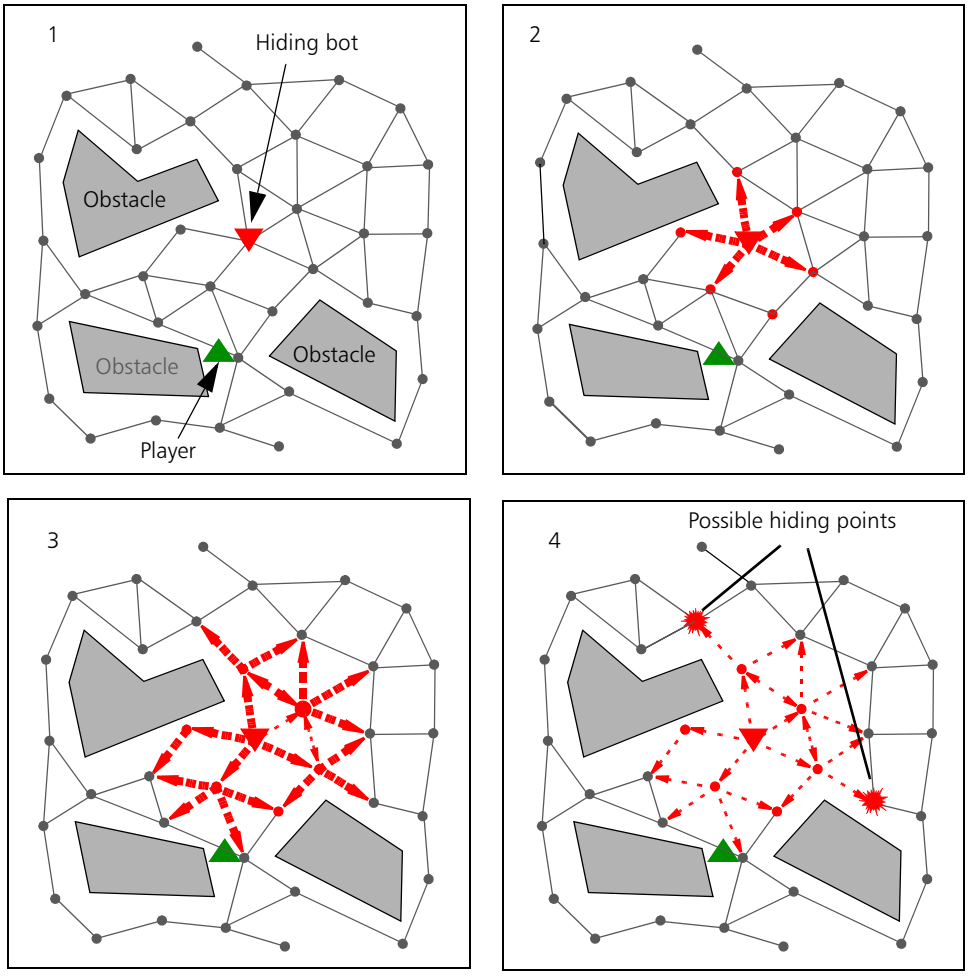


Figure 2-7. Dynamic topological analysis

2.5. B-HAVE Module Tasks

This section describes the models used for the following B-HAVE Module tasks:

- Move To
- Wander
- Flee and Hide.

All B-HAVE Module behaviors use a perception - decision - action model, as follows:

- Perception: The B-HAVE Module determines the entity state, external simulation data, such as enemies detected and sensor data, and topological information.
- Decision: The B-HAVE Module determines the appropriate behavior for the entity and translates that into tasks.
- Action: The B-HAVE Module supplies an intended speed and direction for the entity.

2.5.1. The Move To Task

B-HAVE movement replaces the standard VR-Forces Move to Location and Move to Waypoint tasks. For the B-HAVE Module Move To task, you supply a destination. The entity computes its path (path finding), and follows the path (path following) to its destination. It avoids entities (dynamic avoidance).

A lifeform can walk on pavement, go inside buildings, and go up or down a flight of stairs or a ladder to reach its goal. A car can follow roads to its destination. A tank computes an optimized path, using roads when possible, and going off-road otherwise, to optimize time.

2.5.2. The Wander Task

An entity given the B-HAVE Module Wander task identifies destinations in its vicinity, chooses one randomly and moves towards it. When it reaches its destination, the entity finds a new destination point and repeats the process.

Dozens of lifeform entities can be tasked to wander, providing background human traffic in a simulation. As with any B-HAVE Module task, path finding takes place in the 3D topology, so entities can wander indoors or outdoors.

2.5.3. The Flee and Hide Tasks

The B-HAVE Module offers two more complex behaviors based on spatial reasoning:

- Flee: this behavior allows an entity to flee a danger point or hostile entity and stop only when it is hidden or a specified distance from the danger. The flight behavior is reactive; the Flee task regularly recalculates a safe destination point.
- Hide: this task allows an entity to hide from a group of dangerous entities.

2.6. Traffic Generation (Pattern of Life)

An important use for the B-HAVE Module is to populate simulations with background entity activity, such as vehicular or pedestrian traffic. The Wander task causes entities to wander randomly in an area, but this is somewhat unrealistic because traffic usually moves from one place to another, it does not wander aimlessly. The pattern of life feature creates more realistic traffic patterns by having entities move randomly from spawn points (where they get created), to sink points (where they are deleted). This provides the appearance of intentional traffic patterns.

You can have multiple spawn and sink points to introduce multiple traffic patterns. Vehicles choose a sink point at random. You can configure the speed of entities, the rate at which they are created, and other parameters.

You can also write custom plans to replace the default plan (move to a sink point). Custom plans can include any of the tasks, sets, and global commands available to entity plans.

For an example of using traffic generation, run the SubwayPOL scenario or any of the the FirstExperience scenarios in *./userData/scenarios/B-HAVE_Demos*.

For more information, please see Chapter 5, *Generating Traffic*.

2.7. Advanced Planning Capabilities

B-HAVE Module tasks provide a level of advanced behaviors that are accessible through the VR-Forces Task menu and in the Plan Editor and do not require any programming.

3. Generating PathData

This chapter explains how to generate PathData for use with the B-HAVE Module.

3.1. Introduction

To use the B-HAVE Module tasks in VR-Forces, you must use a terrain that has PathData generated for the areas in which you want to assign B-HAVE Module tasks. You generate PathData in the VR-Forces front-end.

When you generate PathData, the B-HAVE Module creates separate sets of PathData, at various levels of detail, for lifeforms and for ground platforms.

PathData is generated as follows:

1. Load the terrain on which you want to generate PathData or load a scenario that uses that terrain.
2. Optionally, change the PathData configuration.
3. Generate the PathData.
4. Start a new scenario or save and reload the existing scenario.

3.1.1. Creating PathData from Road Vector Networks

To correctly convert road vector networks to PathData, the vector networks should have the following characteristics:

- Segments should be grouped into “edges” whenever possible. An edge is a sequence of segments with no junction.
- Edges should end at junctions, and the endpoints of the connecting roads should be close to each other (less than a few meters apart, and less than the road width).
- Short segments should be avoided, even more so for segments leading to intersections. The recommended size is in the 10-100m range.
- To allow multiple road levels, each road edge should have at least one point with a single altitude candidate, so that this altitude can be propagated to other points in the edge (those points for which the altitude has to be guessed among several candidates).

3.2. Configuring Generation of PathData

The B-HAVE Module provides default values for generating PathData that should work for most terrains. If you want to, you can change the configuration. The B-HAVE Module remembers the most recent PathData configuration and continues to use it until you change it or revert to the default values.

To configure PathData generation:

1. Choose **B-HAVE** → **Configure PathData**. The Generation Options dialog box opens.
2. Select the Generation Options page (Figure 3-1).

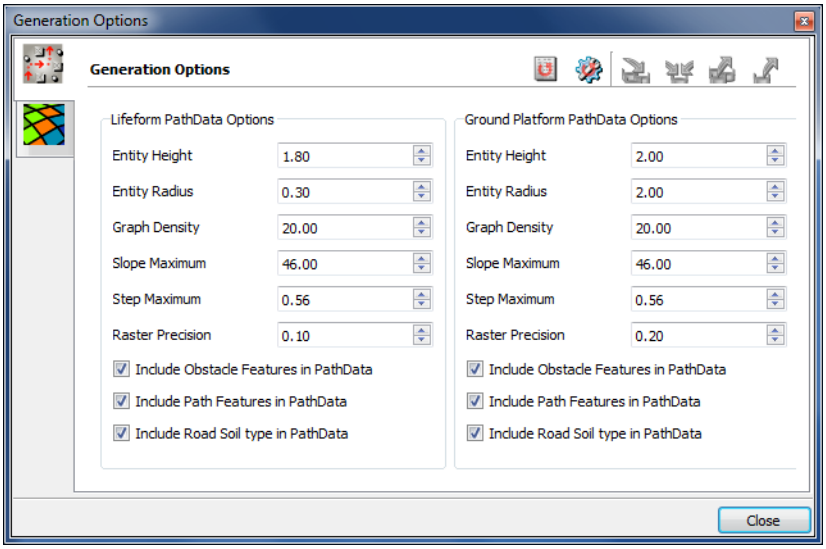


Figure 3-1. Generation Options page

3. On the Generation tab, change configuration values for lifeforms or ground platforms as desired. Table 3-1 describes the parameters.

Table 3-1: PathData configuration parameters

Parameter	Description
Entity Height	The maximum height for this type of entity. Used to calculate the ability of entities to move through doors, under bridges, and so on.
Entity Radius	The maximum radius of the entity bounding box. Used to calculate the width of spaces an entity can move through.
Graph Density	The density of the PathData. Higher density provides higher quality entity movement, but increases processing time.
Slope Maximum	The maximum slope that an entity can move on.

Table 3-1: PathData configuration parameters

Parameter	Description
Step Maximum	The maximum height of steps that an entity can move up. Also the maximum break in the terrain that an entity can cross.
Raster Precision	The precision of calculations for points in the terrain.
Generate Vector Network	When enabled, include vector features labeled as roads in the PathData.

4. On the Soil Type page (Figure 3-2), select or clear soil types as desired. If a soil type is not selected, the B-HAVE Module does not generate PathData for that soil type and B-HAVE-enabled entities will avoid locations with that soil type.

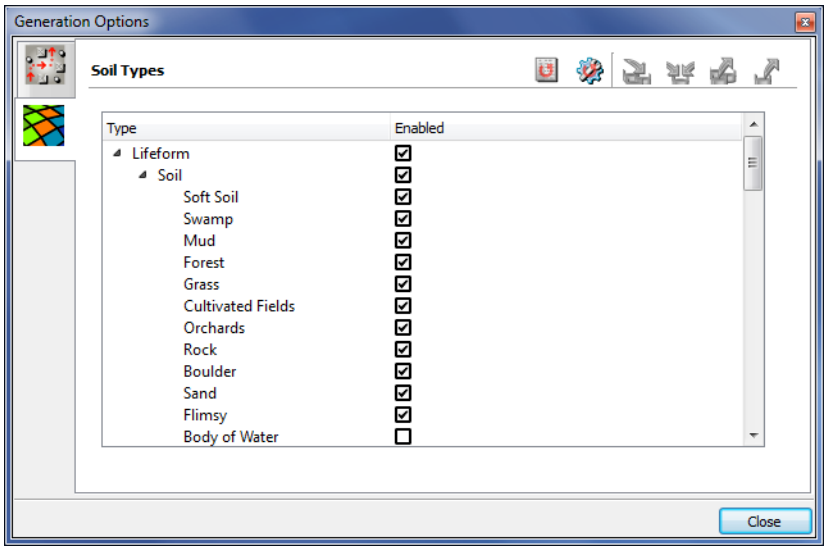


Figure 3-2. Soil Type page

5. Click Close.

3.3. Generating PathData

Generating PathData can last from a few minutes to hours to days depending on the terrain complexity, the level of detail, and your computer's processing power.

The location of the PathData files depends on the type of terrain you generate them for. If the terrain that you loaded references a GDB terrain file, the PathData is placed in a directory under the directory that contains the GDB file. If you generate PathData for an MTF terrain that does not reference a GDB file, the PathData is placed in a directory under the directory where the MTF file is located (usually *./userData/terrains*).

The maximum size of the area on which you can generate PathData is 20 KM by 20 KM

To generate PathData:

1. Choose **B-HAVE** → **Generate PathData**. A tab is added to the side of the VR-Forces window.
2. Click the tab. The Generate PathData dialog box opens (Figure 3-3).

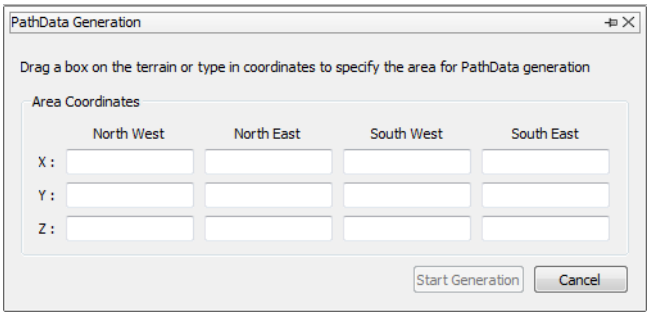


Figure 3-3. Generate PathData dialog box

3. Using the mouse, drag a bounding box around the area in which you want to generate PathData. The coordinates of the area are displayed. You can edit the coordinates by hand if you want to.
4. Click Start Generation. The label on the PathData Generation tab changes to Generating and the Generate PathData command is not available on the B-HAVE menu. You can use VR-Forces for other purposes while PathData is generating. When the operation is complete, VR-Forces displays a message.

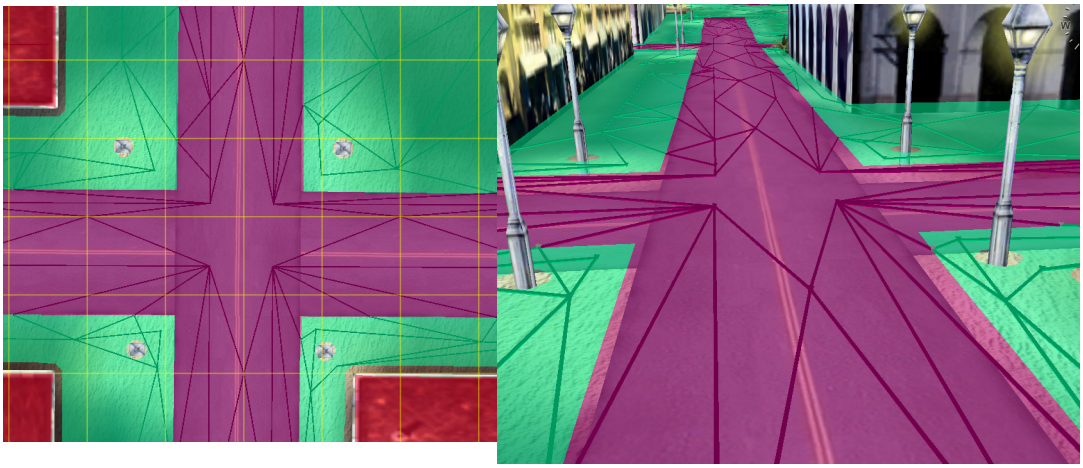
3.3.1. Using Newly Generated PathData

To use newly generated PathData, create a scenario that uses the terrain or open a scenario that uses that terrain. If you generated PathData from an open scenario, you must save the scenario, close it, and then open it again.

3.4. Displaying PathData

You can display the PathData in a terrain for lifeforms or for vehicles. Figure 3-4 illustrates PathData for lifeforms in a portion of the Makland terrain. The lines represent the paths an entity can follow. The lines will be the same for both lifeforms and ground platforms except in areas where a particular type cannot travel. The colors represent different soil types. (The colors are chosen by the underlying Kynapse software and are not configurable.)

You can display PathData in both the 2D and 3D views.



2D 3D

Figure 3-4. PathData display

PathData is not displayed if you are zoomed out too far from the terrain.

➤ To display or hide PathData, choose **B-HAVE** → **Show Generated PathData** → **LifeForm** (or **Ground Platform**). The display of PathData toggles to the opposite of the current setting.

3.5. Troubleshooting

If you do not see the B-HAVE menu, the plug-in was not loaded. Make sure that the file *./plugins/B-Have.xml* is present. If you do not want to load the plug-in, you can temporarily disable loading. To do so, remove *./plugins/B-Have.xml* or change its extension. You can also enable or disable the plug-in in the Plugins Manager in the VR-Forces front-end. (For details, please see Section 5.9, "Managing Plug-ins," in *VR-Forces Users Guide*.)

4. Using the B-HAVE Module in VR-Forces

This chapter explains how to assign B-HAVE Module behaviors to entities. It assumes that you are familiar with VR-Forces and know how to assign independent tasks and how to assign tasks as part of plans.

4.1. Using the B-HAVE Module in a Simulation

When you install the B-HAVE Module, it adds a B-HAVE menu to the menu bar and new commands to the Task and Set menus. You do not have to do any additional configuration to use the B-HAVE Module, other than ensuring that use of plug-ins is enabled.

To use the B-HAVE Module tasks and scripts in a simulation, you must have a terrain with PathData generated for the appropriate type of entity. The procedure for generating PathData is explained in Chapter 3, *Generating PathData*.

4.1.1. Disabling the B-HAVE Module Plug-ins

If, after installing the B-HAVE Module, you do not want to load the plug-ins, you can temporarily disable loading. To do so, remove *./plugins/B-Have.xml* or change its extension. You can also enable or disable the plug-in in the Plugins Manager in the VR-Forces front-end. (For details, please see Section 5.9, "Managing Plug-ins," in *VR-Forces Users Guide*.)

4.2. Creating Multiple Entities

The B-HAVE Module allows you to add multiple entities to a scenario in one operation and to assign a common task to the entities. Entities are added to the scenario as members of a selection group. Creation of multiple entities has the following constraints:

- The entities are added within a specified area. If an appropriate area does not already exist, you can create one as part of the entity creation process.
- If a terrain does not have any PathData associated with it, you can place multiple entities anyplace on the terrain. The entities are placed without regard to terrain features.
- If a terrain has PathData, you cannot place entities on portions of the terrain that do not have PathData. If the area in which you place entities is not entirely within the portion of the terrain that has PathData, the entities are placed only in that part of the area that has PathData.
- If the area you choose for placing entities is too small to contain the number of entities that you specify, some entities might not be created.

When you create multiple entities, VR-Forces automatically puts them in a selection group named group_*number*, where *number* starts at 0 and is incremented for each group created.

In Figure 4-1, Area 1 overlaps the PathData. If, for example, you try to place 25 entities in this area, they are placed only in the portion of Area 1 that overlaps PathData, and only the number of entities that fit in this subset of Area 1 are created.

- If the B-HAVE Module cannot create the number of entities specified (due to one of the reasons mentioned in this section), there is no notification. It is your responsibility to check to see how many entities were created.
- Creating multiple entities as described in this section is not the same thing as pattern of life traffic generation. It is just a way to create a lot of entities at one time and give them a task.

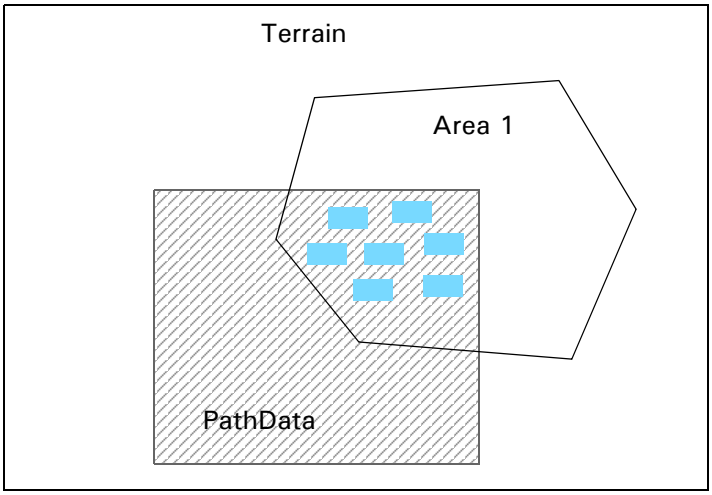


Figure 4-1. Placement of multiple entities

When you create multiple entities you can ask VR-Forces to run an accessibility check before it creates each entity. VR-Forces tries to ensure that the entity being created can reach a specified waypoint from a candidate location in the specified area. This check is primarily designed to avoid putting entities on top of buildings that are unreachable and would not allow the entity to leave the building. It works as follows:

- If the terrain does not have PathData, the accessibility check is disregarded and the entities get created.
- If the terrain has PathData and if the candidate location cannot use PathData to reach the waypoint, VR-Forces does not create the entity at that location. It tries to find another location at which to place the entity. If after multiple attempts to place an entity the accessibility check still fails, VR-Forces proceeds as follows:
  - If this is the first entity that VR-Forces has tried to create, the accessibility check is disregarded and VR-Forces creates the entities (within the constraints previously mentioned).
  - If the accessibility check fails for any entity after the first entity was created, VR-Forces assumes there is no more space in which to create entities and stops trying to create them.

To add multiple entities to a scenario:

1. Choose **B-HAVE** → **Create Multiple Entities**. The Create Multiple Entities dialog box opens (Figure 4-2).

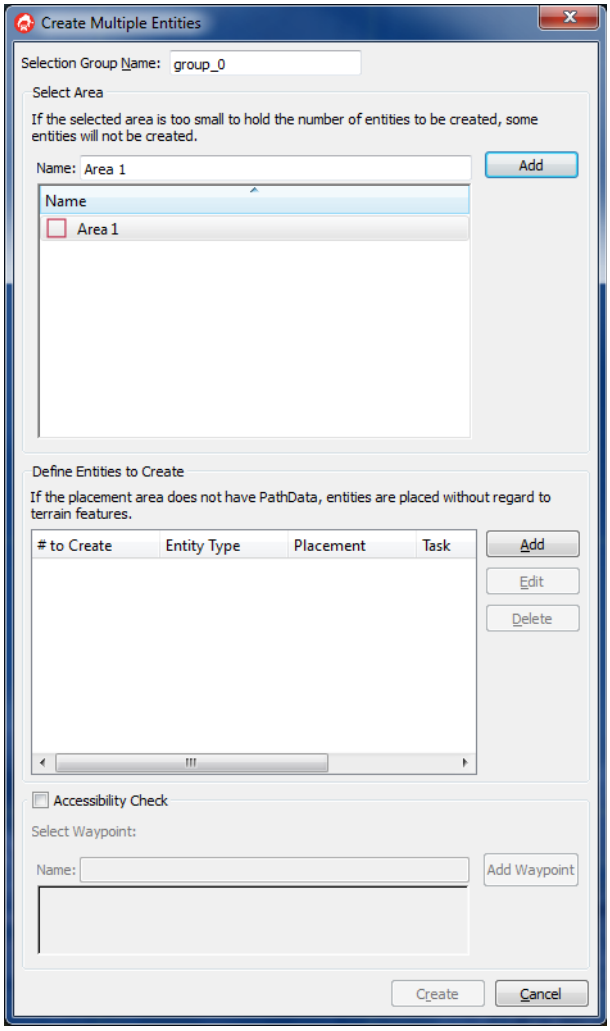


Figure 4-2. Create Multiple Entities dialog box

2. Optionally, change the name of the selection group.
3. In the Select Area group box, type the name of the area in which you want to create the entities, or select the area from the list or on the map. If the area in which you want to create the entities does not exist, add a new area as described in "Adding an Area for Multiple Entity Creation," on page 4-6.
4. In the Define Entities To Create group box, add the entities that you want to create. For details about this part of the procedure, please see "Specifying the Entities to Create," on page 4-6.

5. Optionally, edit or remove any of the entity sets listed in the Define Entities To Create list.
6. Optionally, select the Accessibility Check check box. If you select this check box:
   a. Select a waypoint. You can create a waypoint from this dialog box if necessary.
7. Click Create. The entities are created in the specified area.

4.2.1. Adding an Area for Multiple Entity Creation

To add an area from the Create Multiple Entities dialog box:

1. In the Select Area group box, click Add. A Create Area tab is added to the window.
2. Create an area following the standard VR-Forces procedure for creating areas. For details, please see Chapter 2, *Creating and Placing Objects*, in *VR-Forces Scenario Management Guide*.

4.2.2. Specifying the Entities to Create

When you add multiple entities to a scenario, you can specify any of the supported entity types and assign a common task or set data request to a specific set of entities.

To specify the entities to add in the Create Multiple Entities dialog box:

1. In the Define Entities to Create group box, click Add. The Add Group of Entities dialog box opens (Figure 4-3).

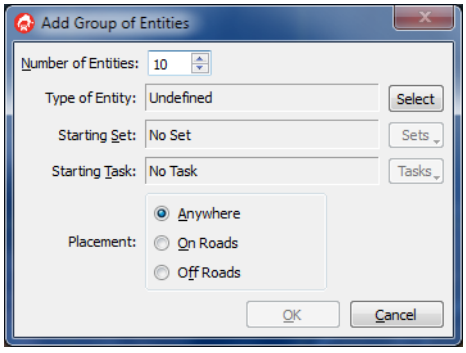


Figure 4-3. Add Group of Entities dialog box

2. In the Number of Entities box, specify the number of entities you want to add for this entity set.
3. Click the Select button that is next to the Type of Entity text box. The Entity TYpe dialog box opens. The list of entity types available is the same list of entities provided on the Entities Palette for ground and lifeform entities.
4. Select the entity type you want to add. The dialog box closes and the entity type is listed in the Type of Entity box.

5. Optionally, specify a starting set data request by clicking the Sets button and selecting a set data request. The list is not context sensitive. You must ensure that the selected set data request is appropriate to the entity type that you are adding.
6. Optionally, specify a starting task by clicking the Tasks button and selecting a task. The list is not context sensitive. You must ensure that the selected task is appropriate to the entity type that you are adding.
7. Optionally, select an option in the Placement group box. VR-Forces will try to place the entities in the desired location.
8. Click OK. The entity set is added to the list in the Define Entities To Create group box (Figure 4-2).
9. Optionally, add additional entity sets.

4.3. B-HAVE Module Tasks

The B-HAVE Module adds several tasks that you access from the Task menu or in plans the same way that you access any other entity task. It also changes how some standard VR-Forces tasks work. The following tasks use B-HAVE movement algorithms instead of the standard VR-Forces tasks when the B-HAVE Module is enabled:

- Follow Entity
- Move to Waypoint (Direct)
- Move to Location (Direct)
- Move Along Route.

When you assign one of these tasks, the B-HAVE Module version is used if:

- The task is applied to an entity type supported by the B-HAVE Module and for which PathData has been generated.
- The task is to be executed on a portion of the terrain that contains PathData (that is, both the entity, and if applicable, the destination are on terrain for which there is PathData).

If these criteria are not met, the native VR-Forces version of the task is used. VR-Forces handles the decision making process for when to use the B-HAVE Module tasks or VR-Forces versions of these tasks. You do not have to do anything except assign the tasks that you want to use. However, you should be aware of these conditions in case your entities behave in unexpected ways.

If, while a B-HAVE Module task is being executed, conditions change so that the criteria for a B-HAVE Module task no longer apply, the B-HAVE Module changes the task to a VR-Forces task. For example, if you give an entity a Move to Waypoint (Direct) task and the entity and waypoint are both inside an area that has PathData for that entity type, the B-HAVE Module assigns a B-HAVE Move to Waypoint (Direct) task. However, if while the entity is moving to the waypoint, you move the waypoint outside of the area with PathData, the B-HAVE Module changes the task to a VR-Forces Move to Waypoint (Direct) task.

When the B-HAVE Module plans a path for an entity, it does not take into account road networks. It just uses the PathData.

- If you give an entity a Move To Waypoint (On Roads) or Move To Location (On Roads) task, the entity uses VR-Forces road following and does not use the B-HAVE Module. If you give an entity a Move To Waypoint (or Location) (Plan Path) task, it uses B-HAVE Module movement for the direct movement portion of the task if the criteria for B-HAVE usage is met.
- B-HAVE tasks in plans from VR-Forces 4.1 or earlier might not be editable. If the Edit button does not become available when you select a task, you will have to delete the task and add it again if you want to change it.

4.3.1. The Flee From Task

The Flee From task causes entities to try to maintain a specified distance from other specified entities (the pursuers). If you select the Stop When Hidden option, an entity stops fleeing when it finds a hiding place, even if the hiding place is in the radius within which it is supposed to flee. This task is continuous unless it is overridden manually or by a conditional test in a plan. The entity will continue to flee any time the distance from the pursuers falls within the specified radius and is in line-of-sight.

- We use the term “pursuers” for convenience. The pursuers do not have to be actively seeking the fleeing entity for this task to be in effect.
- Civilian lifeforms are not configured by default to spot other civilians, only armed entities. This is done for performance reasons, but can be changed in the SMS. Edit the visual-sensor detection-types parameter to change what types of entities are spotted.

To assign a Flee From task:

1. Select the entities that you want to flee.
2. Choose **Task** → **Movement** → **Flee From**. The B-HAVE Flee From dialog box opens.
3. Optionally, filter the list of entities to flee from.
4. Select the entity (the pursuer) from which you want the entities to flee.
5. Specify the distance, in meters, the entities should try to maintain from pursuers.
6. Optionally, to have a fleeing entity stop when it is not within line-of-sight of the pursuers, select Stop When Hidden.
7. Click OK.

4.3.2. The Follow Entity Task

The Follow Entity task works like the standard VR-Forces Follow Entity task except that entities use B-HAVE Module path planning and collision avoidance. To assign this task, follow the procedure in Section 8.28, "Follow Entity,", in *VR-Forces Scenario Management Guide*.

4.3.3. The Hide From Task

The Hide From task causes an entity to hide from specified entities. It is considered hidden when it is not within line-of-sight of the pursuing entities. This task is continuous unless it is overridden manually or by a conditional test in a plan. The entity will continue to hide any time the pursuers are in line-of-sight.

- We use the term "pursuers" for convenience. The pursuers do not have to be actively seeking the hiding entity for this task to be in effect.
- Civilian lifeforms are not configured by default to spot other civilians, only armed entities. This is done for performance reasons, but can be changed in the SMS. Edit the visual-sensor detection-types parameter to change what types of entities are spotted.

To assign a Hide From task:

1. Select the entities that you want to hide.
2. Choose **Task** → **Movement** → **Hide From**. The Hide From dialog box opens.
3. Optionally, filter the list of entities.
4. Select the entities (pursuers) from which you want the entities to hide.
5. Click OK.

4.3.4. The Move to Waypoint (Direct) Task

The Move To Waypoint task sends an entity to the specified waypoint or object using B-HAVE Module path planning and collision avoidance. To assign this task, follow the procedure in Section 8.43, "Move To Waypoint (Direct)," in *VR-Forces Scenario Management Guide*.

4.3.5. The Move to Location (Direct) Task

The Move to Location task sends an entity to the specified location using B-HAVE Module path planning and collision avoidance. You can specify locations that are inside buildings. To assign this task, follow the procedure in Section 8.40, "Move to Location (Direct)," in *VR-Forces Scenario Management Guide*.

4.3.6. The Move Along Route Task

The Move Along Route task is identical to the VR-Forces Move Along Route task except that entities use B-HAVE Module path planning and collision avoidance. If you select Treat Route as Road, VR-Forces uses the road following behavior and does not use B-HAVE Module path planning. To assign this task, follow the procedure in Section 8.36, "Move Along Route," in *VR-Forces Scenario Management Guide*.

4.3.7. The Wander Task

The Wander task causes an entity to move to a series of randomly chosen locations for the specified amount of time. You can constrain wandering to a specified area. If the entity is using a preferred movement mode and there is preferred road data for the terrain, the Wander task will only choose destinations that meet the preference criteria.

To assign a Wander task:

1. Select the entity.
2. Choose **Task** → **Movement** → **Wander**. The Wander dialog box opens.
3. In the Duration group box, select Indefinitely or Timed. If you select Timed, specify the amount of time to wander in hours, minutes, and seconds.
4. In the Movement group box, select Free or Restricted To Area. If you select Restricted To Area, type the name of an area or select it in the list or on the map.
5. Click OK.

4.4. The Set Movement Mode Set Data Request

The B-HAVE Module adds a set data request. Set Movement Mode allows you to specify that when an entity plans its path it should take into account road data, if it is available. This set data request is meaningful only if one or both of the movement modes was enabled when the PathData for this terrain was configured.

When an entity is set to Prefer Roads, the entity gives preference to all segments along the preferred graph over those on the non-preferred movement graph. This is typically used to have vehicles prefer movement along roads, rather than cutting across open terrain, even if the distance along roads is longer.

If the movement mode is Avoid Roads, entities try not to move along roads. If they must cross a road, they take the shortest path across the road.

You can configure an entity to start up in a movement mode or in free movement mode by editing the **initial-movement-mode** parameter in the **kynapse-controller** in the system definition file for this entity type.

To set the B-HAVE Module movement mode for an entity:

1. Select the entity.
2. Choose **Set** → **Action** → **B-HAVE Movement Mode**. The Set Movement Mode dialog box opens.
3. To have the entity take road data into account, select Prefer Roads or Avoid Roads from the Movement Mode list. To have an entity plan its path as if there were no road data, select Free.
4. Click OK.

4.5. Viewing B-HAVE Module Data (Debug Lines)

When B-HAVE-enabled entities are given tasks in VR-Forces, they publish information indicating the path that was chosen for movement. This can be useful while testing a scenario, to check the paths chosen by entities, their line of sight with enemies, and so on.

! Displaying debug lines degrades performance.

Figure 4-4 illustrates 2D debug lines. Figure 4-5 illustrates 3D debug lines. An entity will not move directly along the indicated path because of the runtime path smoothing (described in "Path Following," on page 2-5).

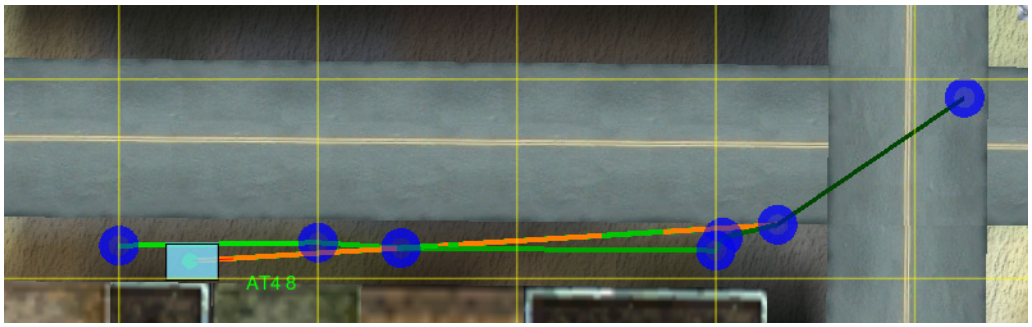


Figure 4-4. B-HAVE debug lines (2D)

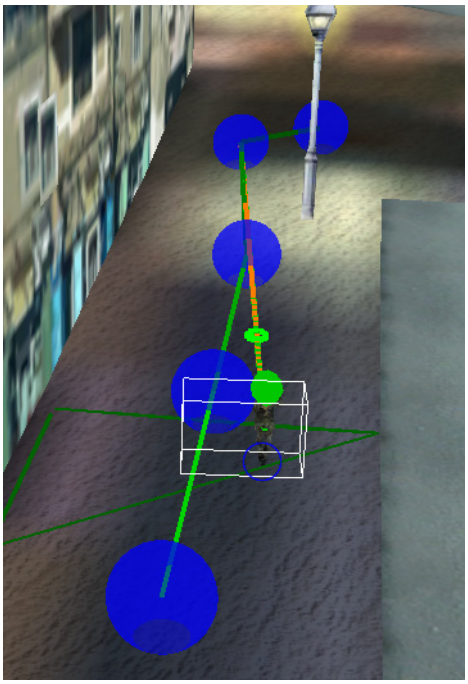


Figure 4-5. B-HAVE paths (3D)

➤ To hide or display debug lines, choose **B-HAVE** → **Show B-HAVE Debug Lines**.

4.6. Configuring an Entity's Soil Type Preferences

The B-HAVE Set Movement Mode set data request ("The Set Movement Mode Set Data Request," on page 4-11) lets you set an entity's movement preference at a very general level. However, there may be cases where you want to more finely tune an entity's choice of where to move. For example, you might set a lifeform entity to avoid roads, but if it has to cross a road, you want it to cross in a crosswalk. Or, you might want entities walking through a park to stay off the grass. To support this type of behavior, you can configure entities to prefer some soil types over others. Naturally, this configuration is useful only if your terrain has detailed soil types.

To configure soil type preferences, add a **path-soil-factors** block (or edit an existing block) to the **kynapse-controller** block in the movement system definition file for an entity type that supports B-HAVE.

Each entry in a **path-soil-factors** block lists a **soil-name** and a **path-factor**. A **soil-name** is a soil type that is listed in a *surfChar.map* file. This file, if present, is specific to each terrain. For an example, see *./data/terrain/Makland/Geometry/OpenFlight/surfChar.map*. For more information about this file, please see Section 13.11, "Converting OpenFlight File Texture Data," in *VR-Forces Configuration Guide*.

The **path-factor** is a preference factor that is applied to the specified soil type. By default, all soil types have a factor of 1.0. B-HAVE plans a path based on the shortest distance to its objective. Increasing the **path-factor** increases the "cost", in distance, of traversing that soil type. A **path-factor** of 2.0 causes an entity to treat travel over that soil type as if it is twice the distance of a soil with a **path-factor** of 1.0. A factor of less than 1.0 creates a preference for that soil type over default soil types. A factor of 0.5 causes an entity to treat travel over that soil type as if it is half the distance of a default soil.

Specifying any factors that are very small or very large may cause path planning to take significantly longer. It is recommended that you keep the path factors between 0.2 and 5.0, and that you minimize the percent of the terrain polygons that have soil factors other than 1.0.

The following **path-soil-factors** list is from *human-bhave.sysdef*.

```
(path-soil-factors
  (item
    (soil-name "sand")
    (path-factor 2.0)
  )
  (item
    (soil-name "grass")
    (path-factor 2.0)
  )
)
```

4.7. Example Scenarios

The B-HAVE Module includes the following example scenarios:

- CarBomb: Entity behavior is controlled through B-HAVE tasks in entity plans. Civilians wander around the business district of a small town. Enemy combatants move through tunnels and buildings to set up positions while a car drives to a location and explodes. Surviving civilians flee. Dismounted infantry search for the enemy combatants. Reinforcements arrive by helicopter, disembark, and help search for enemy combatants or take up positions to secure the town. Civilians return to the bomb site. This scenario uses embarkation. If you are using HLA, you must use RPR FOM 2.0 draft 17.
- CarBomb_NoEmbarkation: This version of the CarBomb scenario does not use embarkation, so you can use it with RPR FOM 1.0. (Embarkation requires use of RPR FOM 2.0, draft 17.)
- Crowd: 100 civilians wander randomly around the city in the TerraSim_SampleUrban terrain. They go in and out of buildings and to multiple floors of buildings.
- FirstExperienceBurstTraffic, FirstExperienceSimpleTraffic, and FirstExperience-CustomPlan. Three scenarios that match the traffic tutorials in B-HAVE First Experience Guide. They demonstrate progressively more complex use of the pattern of life feature.
- Interior Movement: Demonstrates the movement of lifeform entities within buildings. Some civilians wander randomly in and out of buildings, while some DI and other civilians follow pre-determined plans to reach points on different floors of some of the buildings in TerraSim_SampleUrban. DI 1 has no plan. You can task him to move to some of the waypoints to demonstrate dynamic path planning.
- Makland B-HAVE: Mixed vehicle and life form activity on the Makland terrain. Some civilian traffic moves along the roads. Some DIs walk in and out of the palace, changing guard duty, and some military vehicles leave the palace to drive along the roads to locations in the town.
- RaidDemo: This scenario simulates a warehouse raid in Honolulu, Hawaii. Three groups of soldiers approach the raid location by helicopter and boat. They disembark, and four two-man teams move through cover to the doors of the warehouse, while the rest of the soldiers stay back to cover the approach. After the soldiers disembark, the helicopters leave and fly a route nearby, waiting for the signal to come back.

  Once the raid teams are in position, two teams perform a room clear action, while the other two stay covering the exits.

  After the room is cleared, the helicopters are called back. The raid teams fall back, the soldiers embark by groups, and the helicopters and boat make their way back to the carrier they were launched from.

This scenario uses the following scripted tasks:

- Move in Cover. This task moves an aggregate of soldiers through phase lines representing cover.
- Synchronize. This task listens for a specific message from a selected group of entities, and broadcasts a response.
- Move In Parallel. This aggregate task re-tasks each subordinate with a move to, bypassing the aggregate's formation movement.
- Clear Room. This aggregate task moves each subordinate simultaneously along each of two selected routes.

- ReactToExplosion: In this scenario, a bomb in a duffle bag explodes. All civilians in the area react by running from the explosion. A nearby officer radios to all emergency vehicles in the area. The responders then rush to the area, with civilian vehicles pulling over for them as they go.

  This scenario uses the following reactive tasks:

  - Flee from Explosion. Civilians react to the nearby explosion by fleeing in the opposite direction.
  - Make Way for Emergency. Civilian vehicles react to nearby emergency vehicles by pulling to the side of the road. They merge back into the road when the vehicle has passed.

- Scatter. An opposing force entity walks through town. Civilians hide from it.
- SubwayPOL. This scenario generates civilian POL (Pattern of Life) traffic entering and leaving a subway station. The goal is to generate a scene which can be used to stimulate a security camera. The location of the camera is on the rooftop of a building across the road from the subway station, it is marked by the Point-of-Interest tactical graphic.

  In this scenario several patterns are used:

  - Bursts of people leaving the station every 5 minuts when a train comes.
  - Continuous, but slow stream of people entering the station from multiple directions.
  - Pedestrians bypassing the station.

  The scenario requires accces to VR-TheWorld (VR-TheWorld.com) to load streaming terrain.

5. Generating Traffic

This chapter explains how to use the B-HAVE Module to automatically generate entity traffic.

5.1. Generating Traffic

The B-HAVE Module traffic feature (also called spawn and sink or pattern of life) lets you automatically generate background traffic for your scenario.

Automatically generated (spawned) entities are created at spawn points. The default behavior for a spawned entity is to randomly choose a destination, called a sink point, and move towards that point. When it arrives at the sink point it is removed from the scenario. This creates the effect of purposeful activity in an area of the terrain without the need to create and task entities individually.

As an alternative to the default behavior, you can write a plan that is assigned to each entity created at a particular type of spawn point. This plan can include any tasks and set data requests that are appropriate for the entity type. You would typically remove the entity from the scenario when the plan is complete.

The behavior of a spawned entity, as well as other details such as how frequently they are spawned and how many spawned entities can exist at one time, is configured in a "spawn pattern". VR-Forces comes with several spawn patterns to get you started. You can edit spawn patterns and create as many new ones as you wish. Spawn points and sink points for new and edited spawn patterns are dynamically added to the Create menu and Tactical Graphics Palette so that it is easy to create new spawn and sink points for all of the available patterns.

Spawn and sink points are automatically added to the Create menu. However, they are not flagged as Favorites on the Tactical Graphics Palette. If you flag a spawn or sink point as a Favorite (by clicking the star next to the point's name in the list of points) a duplicate entry is added to the Create menu.

5.2. Creating Spawn Points

You can create spawn points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 5-1 illustrates a spawn point.

Figure 5-1. Spawn point

If the plan for a spawn point specifies Use Roads and a spawn point is placed too far away from the road network to use roads, a message is sent to the spawn point's console. If warning icons are enabled, a warning icon is displayed. For details about the Use Roads option, please see "Creating a Spawn Pattern," on page 5-8. For details about configuring console messages and warning icons, please see Section 7.12, "Configuring Object Console Messages," in *VR-Forces Users Guide*.

To create a spawn point from the Create menu:

1. Choose **Create** → **Spawn Points** → *type*, where *type* is one of the spawn patterns. The cursor changes to the waypoint symbol.
2. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
3. Right-click to exit create mode.

To create a spawn point from the Spawn Pattern Manager:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).

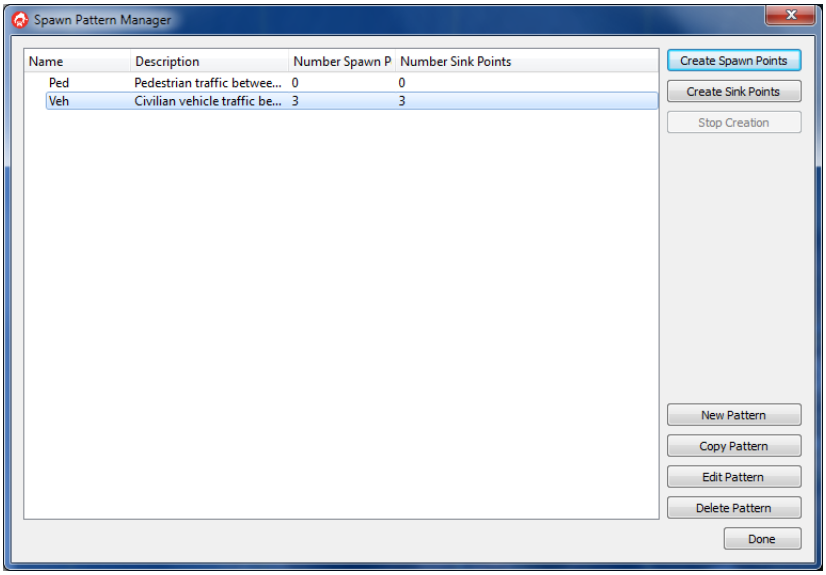


Figure 5-2. Spawn Pattern Manager

2. Select the spawn pattern for which you want to create spawn points.
3. Click Create Spawn Points. The cursor changes to the waypoint symbol.
4. Click on the terrain where you want to place the spawn point. You can continue clicking to create additional spawn points.
5. Right-click or click Stop Creation to exit create mode.

5.3. Creating Sink Points

If you are using the default spawned entity behavior – pick a sink point and move to it – then you have to create one or more sink points for the entities to move to. Entities created at a given type of spawn point automatically choose among sink points of the same type. (You can configure spawned entities to move to any kind of point by editing the spawn pattern. For details, please see "Creating a Spawn Pattern," on page 5-8.) You can create sink points from the Create menu, the Tactical Graphics Palette, or from the Spawn Pattern Manager. Figure 5-3 illustrates a sink point.

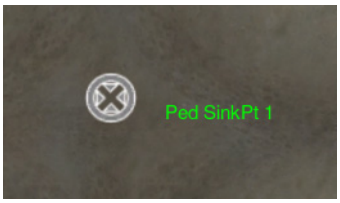


Figure 5-3. Sink point

To create a sink point from the Create menu:

1. Choose **Create** → **Sink Points** → *type*, where *type* is one of the spawn patterns. The cursor changes to the waypoint symbol.
2. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
3. Right-click to exit create mode.

To create a sink point from the Spawn Pattern Manager:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).
2. Select the spawn pattern for which you want to create sink points.
3. Click Create Sink Points. The cursor changes to the waypoint symbol.
4. Click on the terrain where you want to place the sink point. You can continue clicking to create additional sink points.
5. Right-click or click Stop Creation to exit create mode.

5.4. Spawn Patterns

A spawn pattern configures the following aspects of spawned entity creation and behavior:

- The name and description of the spawn pattern.
- The timing of entity creation.
- The type of entity created.
- The spawned entity's plan.

A spawn pattern's name identifies it on the Create menu and is used to build the label for individual spawn points, for example Ped SpawnPt 1, Ped SpawnPt 2, and so on.

A scenario can have multiple spawn patterns and each spawn pattern can spawn one or more entity types. Spawning multiple entity types in one pattern is useful when you want heterogeneous traffic. For example, in a stream of pedestrians you might want to include men, women, and children. In other cases you might want a specific entity type engaging in specific behaviors and would only spawn one entity type in that pattern.

If you use the default spawned entity behavior, you can configure entity movement, as follows:

- Maximum and minimum speed. VR-Forces will randomly assign a speed between the maximum and minimum settings. It will not accept a speed that is greater than the maximum speed for the entity in the OPD file.
- Use Roads. Enables or disables road-following behavior for vehicles. Road following behavior is described in Section 3.5, "Collision, Obstacle, and Feature Avoidance," and Section 7.5, "Entity Movement On Roads and Off Roads," in *VR-Forces Scenario Management Guide*.

When you use the default plan, the Create Spawn Pattern dialog box lists the sink points that this pattern will use.

5.4.1. How Spawn Events are Scheduled

Spawn patterns control the time period during which entities are spawned and the rate at which they are spawned during that time period. A spawn pattern can use one of the following options:

- Continuously. Entities are spawned as soon as the scenario starts and throughout its life.
- Time of Day. Entities are spawned between a starting time of day and ending time of day.
- Simulation Time. Entities are spawned between a specific starting and ending simulation time.

Figure 5-4 illustrates the difference between continuous and timed spawning.

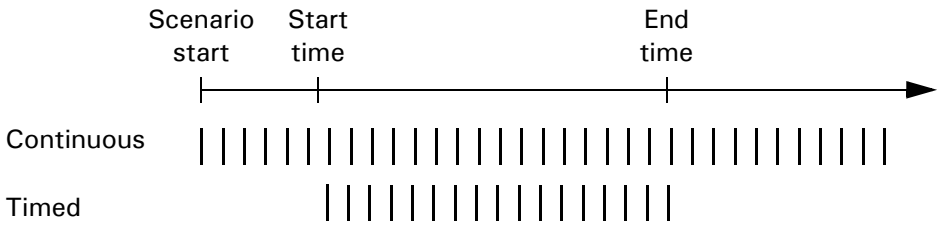


Figure 5-4. Continuous spawning compared to time-based

After you specify a time option, you must specify a spawn interval and a spawn interval variance. The spawn interval determines how frequently entities are spawned. The spawn interval variance introduces some variability to the interval so that there is a more random and naturalistic flow of entity creation.

For example, if the spawn interval is 10 seconds, an entity is spawned every 10 seconds, plus or minus the variance. The variance is a percentage of the spawn interval. If the variance is 10%, then given a spawn interval of 10 seconds, an entity is spawned every 9 to 11 seconds. The spawn intervals are not fixed to an absolute time scale. A new spawn interval starts each time a spawn event occurs.

Figure 5-5 illustrates a series of spawn events with a spawn interval of 10 seconds and a spawn interval variance of 0% and a series of hypothetical events with a spawn interval of 10 seconds and a spawn interval variance of 40%. In the second case, each spawn event can take place anywhere between 6 and 14 seconds after the previous event. Notice that a new spawn interval starts at each spawn event.

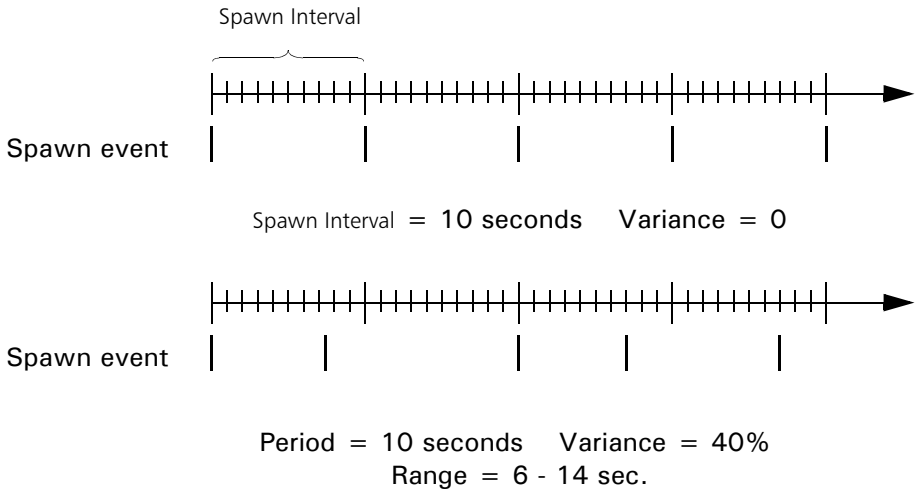


Figure 5-5. Spawn period and variance

The frequency with which a spawn point spawns entities can be affected by the total number of entities that it has spawned. If the maximum number of simultaneous entities for a spawn point is reached, no more entities are spawned from that point until entities from that spawn point are removed from the scenario.

5.4.2. Spawning Entities in Bursts

You can introduce additional variation to the spawn frequency by specifying that entities be spawned in bursts. A burst pattern controls spawning behavior within a fixed period of time. For example, you could use a burst pattern to represent a cycle of train disembarkations or rush hour traffic.

Burst patterns have the following parameters:

- Burst Period. The period of time during which a burst can take place.
- Burst Length. The percentage of the burst period in which to spawn entities.
- Offset Start Time. An offset from the start of the burst period at which to start spawning entities.

Within a burst, entities are created at the rate determined by the spawn interval and spawn interval variance values.

Figure 5-6 illustrates a series of bursts. The burst period is 10 minutes. The offset start time is two minutes, so each burst starts two minutes after the start of the burst period. The burst lasts for 40% of the length of the burst period, in this case four minutes. Within each burst, the frequency of spawn events is controlled by the spawn interval and variance values.

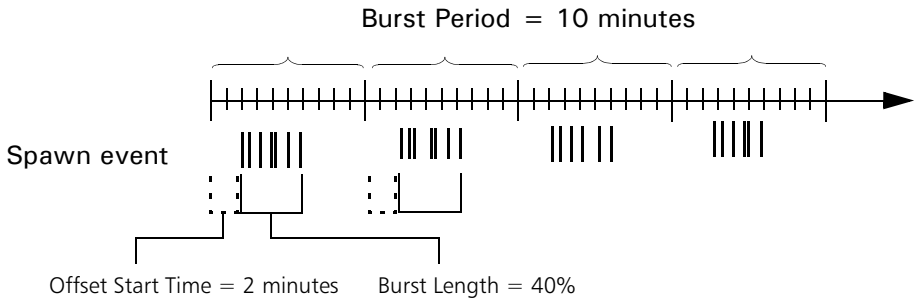


Figure 5-6. Burst periods

5.4.3. Using a Custom Plan

As noted in "Generating Traffic," on page 5-2, the default behavior for a spawned entity is to choose a sink point, move to the sink point, and be removed from the scenario. However, you can introduce more complex behaviors for spawned entities by writing a custom plan for a spawn pattern. By creating a custom plan, the entities can engage in a complex set of behaviors, but you still have the benefit of automatic generation. An example of using a custom plan would be to generate a steady stream of jets taxiing to a runway and taking off from an airport.

Custom plans can include all of the tasks, sets, conditions, and global commands that a regular entity plan has. The plan is treated as an individual entity plan and applied to each entity as it is created. When you write a custom plan, entities do not move to sink points. They remain in the scenario until their plan removes them. You can delete entities without needing to specify the specific entity with the Delete Self global command.

5.4.4. Default Spawn Patterns and Scenario-Specific Patterns

The B-HAVE Module includes default spawn patterns for civilian vehicles and civilian lifeforms. When you create a scenario, these patterns are added to the scenario. When you save a scenario, the spawn patterns get saved with the scenario in the scenario extras file (*.xtr*). (This includes the default spawn patterns if you do not edit them, the edited versions if you have edited them, and any new spawn patterns you have created.) They are now scenario-specific. The next time you load the scenario, it uses the spawn patterns in the scenario extras file, not the default spawn patterns.

If you want to have additional default spawn patterns for your scenarios, you can add entries to *./appData/settings/vrfSim/defaultSpawnTemplates.mtl*. Copy one of the patterns in the file, then change the name and parameter values to suit your needs. Or, create a spawn pattern in a scenario and copy it from the *.xtr* file to *defaultSpawnTemplates.mtl*.

5.5. Creating a Spawn Pattern

Previous sections have explained the various components of a spawn pattern. This section explains how to create one.

To create a spawn pattern:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).
2. Click New Pattern. The Create Spawn Pattern dialog box opens (Figure 5-7).

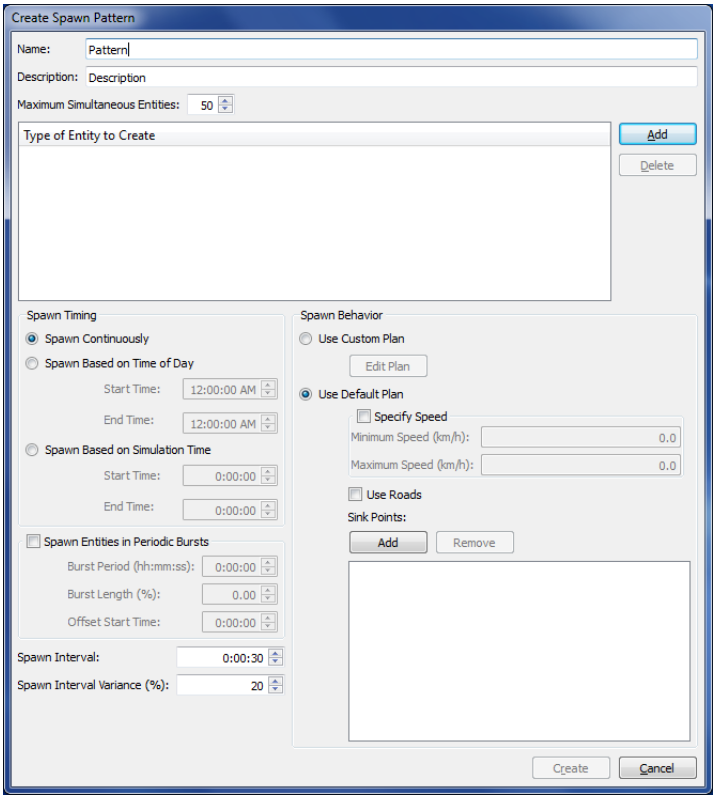


Figure 5-7. Create Spawn Pattern dialog box

3. In the Name box, type a name for the spawn pattern. This name is used for the label for spawn points and sink points created from this pattern. This is a required field.

4. In the Description box, type a description of the pattern. This description is listed in the Spawn Pattern Manager to help identify the purpose of the pattern. This is a required field.

5. Specify the maximum number of entities that can be spawned from a spawn point using this pattern and exist in the scenario at the same time. When the maximum is reached for a given spawn point, no more entities are spawned from that point until an entity that was spawned from that point is removed from the scenario.

   For example, if the maximum number of simultaneous entities for a spawn pattern is 200 and you have 10 spawn points of that type, you could have as many as 2000 spawned entities. None of the spawn points will spawn more than 200 entities at one time.

A high maximum number of simultaneous entities along with a small spawn interval can quickly generate tens to hundreds of entities. This may affect performance.

6. Specify the entity types to create:
   a. Click the Add button. The Entity Type dialog box opens. This dialog box lists entities by category.
   b. Select the category for the entity type you want to add.
   c. Select the entity type that you want to add. The entity is added to the Type of Entity to Create list.
   d. Add additional entity types as desired.

The Entity Type dialog box lets you specify all types of entities, cultural features, and control objects, some of which do not necessarily make any sense to spawn. It is up to you to select types of entities that make sense.

7. Choose an option in the Spawn Timing group box, as follows:
   – Spawn Continuously. Entities are spawned through the simulation.
   – Spawn Based on Time of Day. Specify the starting time of day and ending time of day during which you want entities to be spawned.
   – Spawn Based on Simulation Time. Specify the starting simulation time and ending simulation time during which you want entities to be spawned.
8. Optionally, select Spawn Entities in Periodic Bursts. (For an explanation of periodic bursts, please see "Spawning Entities in Bursts," on page 5-7.) If selected, configure as follows:
   – Burst Period. The time period for each burst.
   – Burst Length. The length of time, as a percentage of the burst period, during which entities are spawned.
   – Offset Start Time. An offset from the start of the burst period to start spawning entities.
9. Specify the spawn interval. This specifies the amount of time between each spawn event, modified by the spawn interval variance.
10. Specify the spawn interval variance as a percentage of the spawn interval.
11. In the Spawn Behavior group box, select a behavior option as follows:
   – Use Custom Plan:
      a. Click Edit Plan. The Edit Custom Plan dialog box opens.

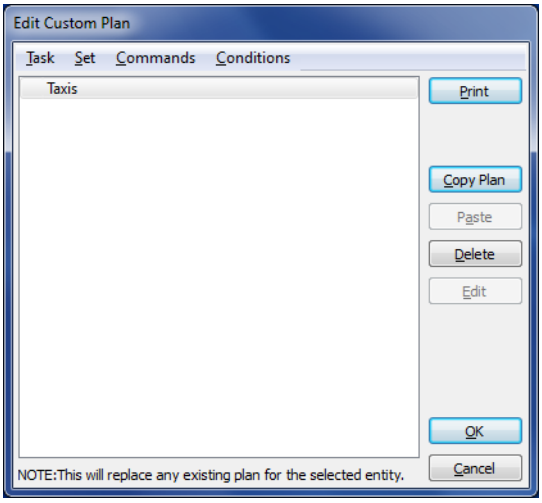


Figure 5-8. Edit Custom Plan dialog box

b. Write a plan that each entity created from this spawn pattern will use. Unless you want the entities to remain in the scenario permanently, end the plan with a Delete Self global command. (For details about how to write plans, please see Chapter 11, *Writing Plans*, in *VR-Forces Scenario Management Guide*.)

– Use Default Plan:

a. Optionally, select the Specify Speed check box and specify the minimum and maximum speeds for spawned entities. If you do not select this option, spawned entities move at the default ordered speed in their OPD file. If you select it, they move at randomly chosen speeds between the minimum and maximum values. This option introduces some variance in behavior. Although the GUI allows you to set any maximum speed, entities will not move faster than the maximum speed in the OPD file.

b. To have an entity move on roads, select the Use Roads check box. If this option is cleared, entities move directly towards sink points without regard to vector road networks.

! Use Roads is valid only for ground vehicles. Do not select this option for any other entity type.

If Use Roads is selected, spawn points must be within 10 meters of the road. If a spawn point is farther away, the entities are removed from the scenario as soon as they are created.

If a sink point is more than 10 meters from a road, entities that are moving towards that sink point move to the point on the road that is closest to it and are then removed from the scenario.

---

c. Optionally, use the Add button to edit the Sink Points list. A new spawn pattern will not have any sink points listed because until you create the spawn pattern, you cannot create any sink points for that pattern. Entities will automatically use sink points that match the spawn pattern, so after you create them, you do not have to add them to the list. However, they can also move to waypoints. If you want spawned entities to move to waypoints, you must add them to the list.

12. Click Create.

5.5.1. Copying a Spawn Pattern

If you want to create a new spawn pattern that will be similar to an existing spawn pattern, you can copy the existing spawn pattern to a new name.

To copy a spawn pattern:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).
2. Select the spawn pattern that you want to copy.
3. Click Copy Pattern. The Create Spawn Pattern dialog box opens. It has a default name based on the original pattern name.
4. Edit the spawn pattern. The meaning of each option is described in "Creating a Spawn Pattern," on page 5-8.
5. Click Create.

5.6. Editing a Spawn Pattern

You can change any of the parameters of a spawn pattern. The changes do not affect any entity that has already been spawned.

To edit a spawn pattern:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).
2. Select the pattern that you want to edit.
3. Click Edit Pattern. The Edit Spawn Pattern dialog box opens (Figure 5-9).

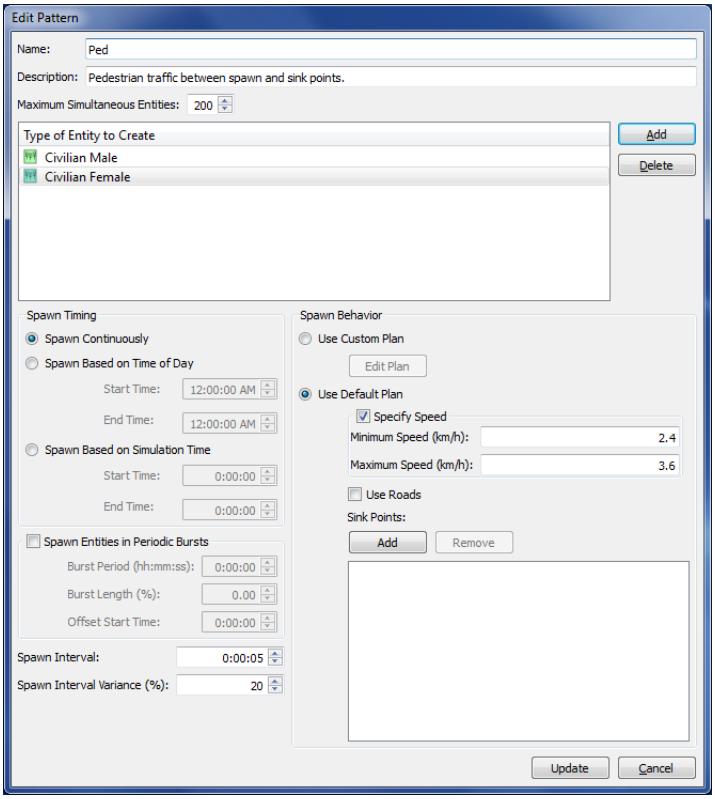


Figure 5-9. Edit Spawn Pattern dialog box

4. Change any value as desired. The meaning of each option is described in "Creating a Spawn Pattern," on page 5-8.
5. Click Update.

5.7. Deleting a Spawn Pattern

If you delete a spawn pattern, all spawn points of that type get deleted. Sink points do not get deleted. The spawn and sink points of this type are removed from the Create menu and Tactical Graphics Palette. If the scenario is running, any entities that were spawned from these points continue executing their plans.

To delete a spawn pattern:

1. Choose **B-HAVE** → **Spawn Pattern Manager**. The Spawn Pattern Manager opens (Figure 5-2).
2. Select the spawn pattern that you want to delete.
3. Click Delete Pattern. You are prompted to confirm the deletion.
4. Click Yes.

6. B-HAVE Messages API

This chapter explains how to use the B-HAVE Messages API to send tasks and set data requests to VR-Forces.

6.1. Introduction

The B-HAVE Module provides an extension to the VR-Forces Remote Control API that lets you access B-HAVE Module functionality. The API allows you to programmatically send messages to create entities and query or modify their behavior during a simulation. For example, through the API, you can assign a new B-HAVE Module task to a specific entity. For information about the VR-Forces Remote Control API, please see *VR-Forces Developers Guide*.

6.2. Sending B-HAVE Tasks

You can use the API to send the following tasks to entities:

- Move to Location (Direct)
- Move to Waypoint (Direct)
- Move Along Route
- Flee
- Hide
- Wander
- Follow Entity.

To assign a task, you create a *DtBhaveTask* object and specify the task name and one or more arguments. The file *bhaveTask.h* contains the definitions of the all the task names available and the task-specific arguments.

The way that you send a task to the B-HAVE Module depends on where the code is executing. If you are issuing the task through the VR-Forces front-end, either by rebuilding the front-end or building a plug-in, you would use code similar to the following example and then emit a signal. This example code tells an entity to hide from two other entities:

```
// Hide from two entities (ent1 and ent2)
DtBhaveTask task;
task.init();
task.setArg1Text(BHAVE_HIDE_TASK);
task.setArg4Text("ent1, ent2");
```

If you are sending the task from an external application, that is, one that is not part of the VR-Forces front-end, you would use the code in the previous example to specify the task and then you would have to send a *DtTaskCommand*, as in the following example:

```
DtTaskCommand cmd;
cmd.setTask(task);
cmd.setObjectName(...);
remoteController->sendTaskMsg(...);
```

6.3. Sending B-HAVE Set Data Requests

As with tasks, you can send a set data request from within the VR-Forces front-end or from an external application. If you send a set data request from the front-end, use the following command syntax:

```
DtSetBhaveMovementMode setMovementMode;
setMovementMode.setMovementType(DtBhaveMovementModePreferRoads);
```

This example sets the movement mode of an entity to Prefer Roads. The different movement modes are defined in *setBhaveMovementMode.h*.

If you send the set data request from an external application, specify the set data request as shown above; then send a *DtSetDataCommand*:

```
DtSetDataCommand cmd;
cmd.setSetRequest(setMovementMode);
cmd.setObjectName(...);
remoteController->sendSetDataMsg(...);
```

6.4. Sending Interface Messages

The API offers different kinds of messages to handle entity creation.

The B-HAVE Module API lets you send the following messages to the simulation engine:

- To create a group of entities, send a DtIfCreateMultipleEntities message.
- To create a selection group, send a DtIfCreateSelectionGroup message.

6.5. Using the API

To use the API, you must first register the different kinds of messages with the remote controller. The *bhaveMsgs.h* file has the list of calls available to automatically register the various kinds of messages supported by the API.

For example, to register the B-HAVE tasks available, use:

```
registerBhaveTasks(controller()->taskFactory());
```

where controller() returns the *DtVrfRemoteController* object associated with your application.

Once the tasks are registered in the controller, you can create *DtBhaveTask* commands and deliver them through the controller. Check the *./examples/bhaveRemoteControl* project for an example illustrating the use of the B-HAVE API inside VR-Forces.

6.6. Building a Project

The B-HAVE Module Remote Control API is based on the VR-Forces Remote Control API. Therefore, if you want to build a project that uses the B-HAVE Module Remote Control API, you must add the libraries from the VR-Forces Remote Control API and the *bhaveMsgsRT[d].lib* file to the link line of your project. The B-HAVE Module API header files are located in *./include/bhave*. For information about how to set up a project that uses the VR-Forces Remote Control API, please see *VR-Forces Developers Guide*.

Index

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

W

Link - Simulate - Visualize

BHV-4.3-1-150217

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com