VR-Exchange

Users Guide

VT MAK
A company of VT Systems

VR-Exchange

Users Guide

Version 2.6

VT MAK
A company of VT Systems

VT MAK
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 VRE-2.6-1-181112

Contents

Preface

Introduction

Installing VR-Exchange

Using VR-Exchange

Working with Broker Connections

Preface

This manual is written for persons who will install or use VR-Exchange. The manual assumes that you are familiar with your operating system and windowing environment.

Please see *VR-Exchange Release Notes* for updates to this manual and the latest information about your version of VR-Exchange.

How the Manual is Organized

This manual is organized as follows:

Chapter 1, *Introduction*, describes VR-Exchange and its architecture. It also contains an important note about the limitations of translation software.

Chapter 2, *Installing VR-Exchange*, describes installation, including setting up the license server and installing an RTI.

Chapter 3, *Using VR-Exchange*, explains how to run VR-Exchange and use it to manage connections and view the data being processed.

Chapter 4, *Working with Broker Connections*, explains how to add, edit, and configure connections.

Chapter 5, *VR-Exchange Tutorials*, shows how to set up a DIS-to-HLA exchange and and RTI-to-RTI exchange.

Appendix A, *Broker Details* describes the configuration parameters for each broker, lists the supported translators, and lists the command-line options that you can use when you start brokers.

The *MAK Products Glossary* defines terms common to real-time simulations and MAK products.

MAK Products

VR-Exchange is a member of the VT MAK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MAK 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.

- MAK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MAK RTI is optimized for high performance. It has an API, RTIspy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you a 2D and 3D views of a simulated environment.

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

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

- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes 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 simulation objects 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.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™ 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.

  - ECOSim. Enhanced Company Operations Simulation (ECOSim) is a company-level training simulation that teaches leaders how best to deploy troops, UAVs, convoys, and other assets. ECOSim focuses on ease-of-use, rapid scenario generation, runtime operator control, and realistic and reactive human simulation.
  - 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.
- RadarFX. RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.
- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.

How to Contact Us

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

Telephone

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

E-mail

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

Internet

MAK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses
Product version and platform information:	www.mak.com/support/product-versions
For the free, unlicensed MAK RTI:	www.mak.com/support/bonus-material

Post

Send postal correspondence to:	VT MAK 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 MAK 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 VR-Exchange home directory. For example, the directory *vrexchange2.6/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

MAK software products may use code from third parties. This section contains summary license information and where required, specific license documentation required by these third parties.

The following libraries are covered by the GNU Lesser General Public License (LGPL) license:

- CIGI
- DevIL
- GEOS
- GStreamer
- LibIconV
- OPCODE
- OpenAL
- OSG
- OsgEarth
- Pthread
- Qt
- Qwt
- LibWebSockets.

The following libraries are covered by the ZLib license:

- Bullet Physics
- Zlib
- LibPNG
- LibSDL
- Minizip.

The following libraries are covered by the MIT license:

- Expat
- FreeGLUT
- GDAL
- GeoTiff
- HarfBuzz
- LibCURL
- LibSquish
- LibXML
- LUA
- LuaBind
- Proj.

The following libraries are covered by the BSD license:

- FreeType
- GLEW
- JPEG
- LibTiff
- Protobuf
- PCRE.

The following libraries are covered by the commercial licenses:

- SpeedTree
- Sundog Triton
- Sundog SilverLining
- FlexLM
- Gameware
- JRM Technologies SenSimRT and SigSimRT
- GLStudio
- OpenFlight
- CDB API.

The following libraries are covered by custom licenses:

- NVidia CG Toolkit
- FreeImage (FreeImage Public License)
- Boost (Boost Software License)
- Poco (Boost Software License).

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLLite is in the public domain.

Boost License

VR-Link, and all MAK 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 MAK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MAK Products. MAK 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/.

1. Introduction

This chapter describes VR-Exchange's features and its architecture.

1.1. The VR-Exchange Architecture

VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, simulations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or simulations using different RTI implementations can interoperate.

VR-Exchange permits simulations to interoperate through the use of a shared memory space (Portal) and brokers (Figure 1-1). Each broker translates between its native protocol and the VR-Exchange common simulation representation. The translated data passes through the Portal and is translated by the other brokers to their protocol.

Given this architecture, it is not necessary to create multiple translation programs that map between specific sets of protocols. You only need to create a broker for a particular protocol, and simulations can interoperate with any other simulations for which a broker exists.

You can run multiple instances of a broker, each configured for a particular simulation. For example, Figure 1-1 shows three different federations. Each uses the broker, but each broker is configured to use different RTIs and different FOMs. Each unique broker configuration is called a connection.

Because the Portal and brokers use shared memory, the Portal and all brokers must run on the same computer.

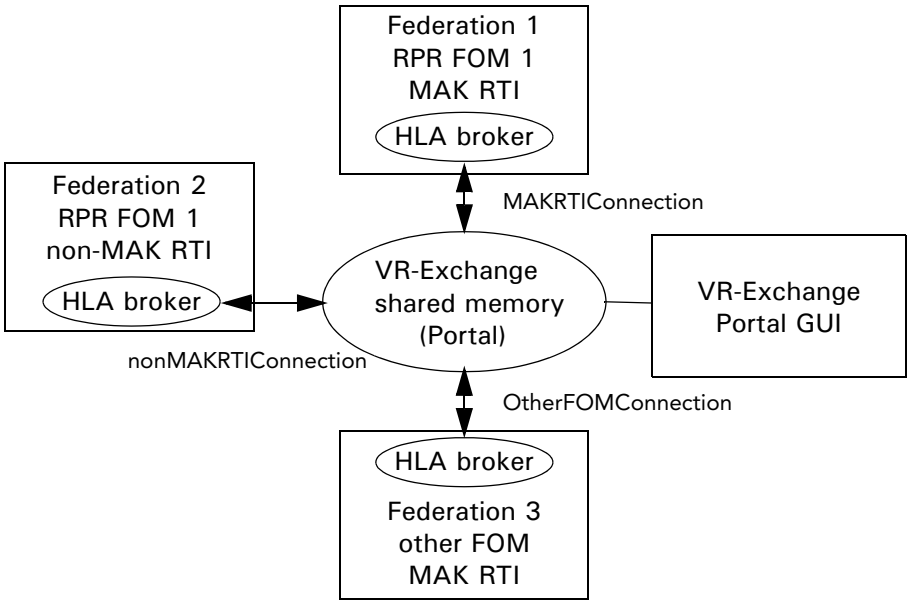

Figure 1-1. VR-Exchange architecture

1.1.1. Translators

Each broker has translators for the object or interaction classes that it supports. Figure 1-2 shows how a translator works. Imagine language-specific HLA FOMs, in this case one for American English and one for British English. Their brokers have translators for the different concepts in the English language. The leftmost federation sends a “lorry” message. The vehicle translator in the broker translates “lorry” to a generic description “wheeled cargo vehicle” and puts the data in the VR-Exchange shared memory area. A second federation, also using the British English FOM, takes the vehicle data from the Portal and translates it back to “lorry”. Finally, a third federation, which uses an American English FOM, reads the vehicle message from the Portal and translates it to its American English equivalent, “truck.”

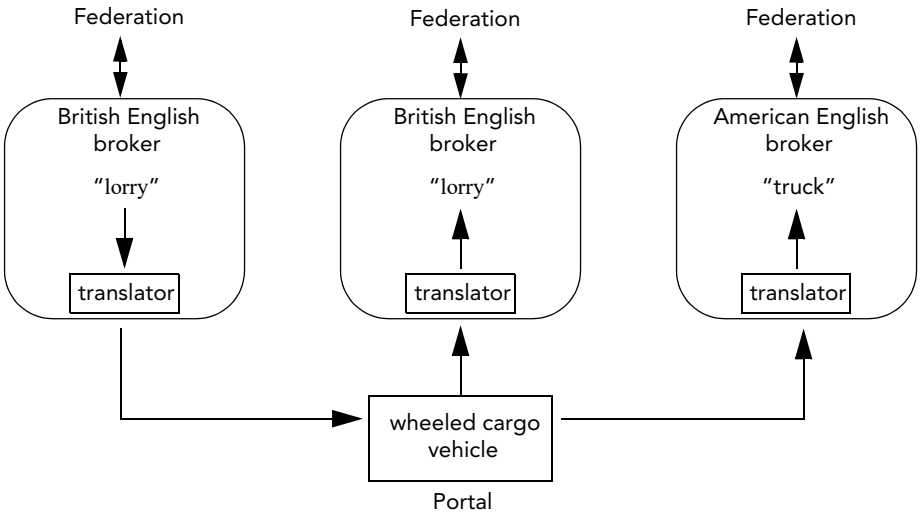

Figure 1-2. How translators work

You can disable translators by editing a connection. Since each connection has its own configuration, you can filter the objects and interactions that the federations are communicating through VR-Exchange.

1.1.2. Using VR-Exchange to Manage a Federation of Federations

In addition to using VR-Exchange to enable heterogeneous federations to interoperate, you can use it to help manage a federation of federations, regardless of whether they use the same FOMs and RTIs. Consider the configuration in Figure 1-3. Four federations are connected via brokers to a VR-Exchange Portal. These Portals are connected to brokers that form their own federation. If any of the individual federations fail, then each instance of VR-Exchange can detect this and drop it from the exercise, while the others can continue.

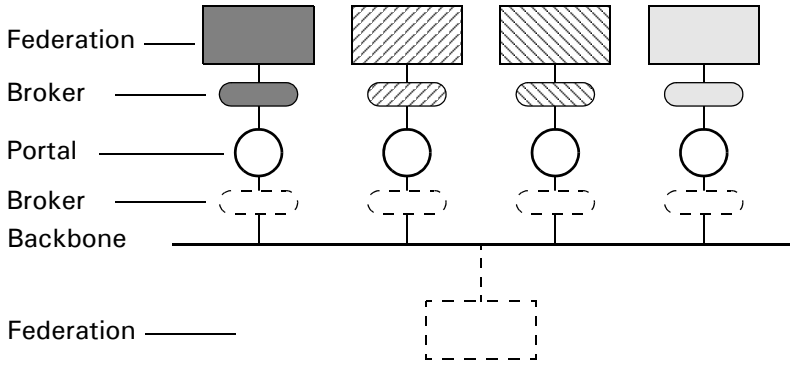

Figure 1-3. Using VR-Exchange over a WAN

1.1.3. The VR-Exchange Portal Application

VR-Exchange has a graphical user interface (Portal application) that provides a view into the shared memory space. It displays the connections in use and the objects and interactions being translated (Figure 1-4). The Portal application does not control the translation process. Translation takes place in the brokers, which are separate programs.

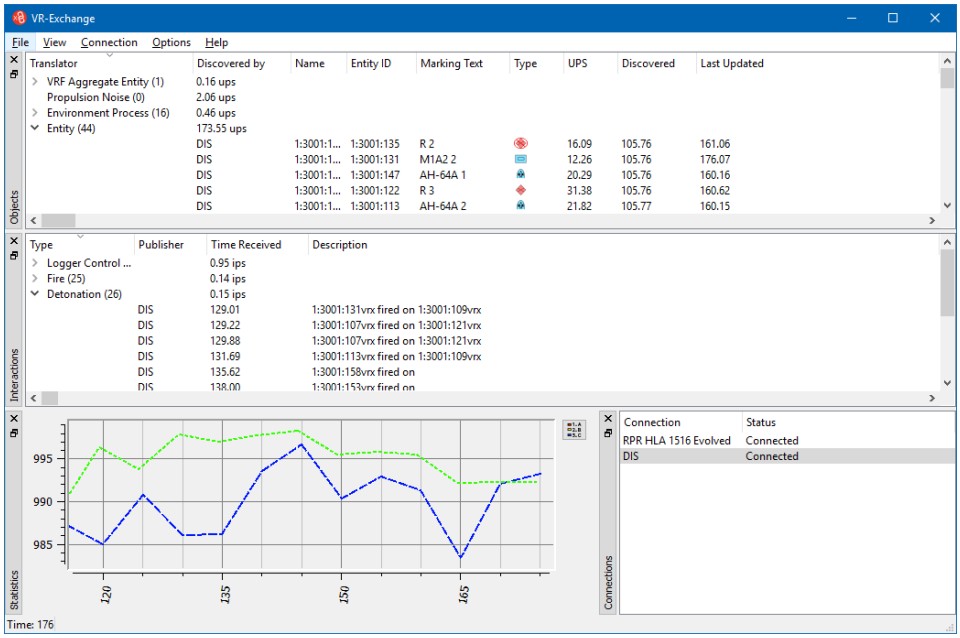

Figure 1-4. VR-Exchange Portal window

1.1.4. FOM-Agility and LROM-Agility

VR-Exchange brokers accommodate different FOMs and LROMs using VR-Link FOM Mappers and LROM Mappers. Mappers are dynamic link libraries that map the objects and concepts of a particular FOM or LROM to the VR-Link internal object model. Since the VR-Exchange brokers are VR-Link applications, they can use these mappers to switch between different FOMs and LROMs. For details about FOM mapping, please see *VR-Link Developers Guide*.

1.2. The VR-Exchange API

The VR-Exchange API has two primary components, the Broker API and the Portal API. The Broker API lets you create brokers for use with VR-Exchange or extend the brokers provided by MAK. The Portal API allows brokers to send messages to and receive updates from, the Portal. For details, please see the class documentation.

1.3. Brokers Supplied

VR-Exchange includes brokers for HLA, DIS, and DDS. It has HLA brokers for the HLA 1.3 specification, for the SISO Standard Dynamic Link Compatible C++ API for IEEE 1516 specification, and for the HLA Evolved (IEEE 1516-2010) specification. Subject to licensing, you can run as many instances of these brokers as you want. The HLA brokers support the versions of the RPR FOM supported by VR-Link, or any FOM that has a VR-Link FOM Mapper.

The DIS broker receives DIS 4, 5, 6, and 7 PDUs. It sends DIS 5 PDUs.

Table 1-1 lists the translation capabilities of each broker. For more information about translators, please see "HLA Translators," on page A-2, "DDS Translators," on page A-12, and "The DIS Broker," on page A-9.

Table 1-1: Broker translators

Translator	HLA	DIS	DDS
Acknowledge	X	X	
Action Request	X	X	
Action Response	X	X	
Active Sonar	X	X	
Additional Passive Activities	X	X	
Aggregate	X	X	
Areal Object	X	X	
Collision	X	X	
Comment	X	X	
Create Entity	X	X	
Data	X	X	
Data Query	X	X	
Designator	X	X	
Detonation	X	X	X
Emitter System	X	X	
Entity State	X	X	X
Environment Process	X	X	
Event Report	X	X	
Fire	X	X	X
Gridded Data	X	X	
IFF	X	X	
Linear Object	X	X	

Table 1-1: Broker translators

Translator	HLA	DIS	DDS
Logger Control	X	X	
Point Object	X	X	
Propulsion Noise	X	X	
Radio Receiver	X	X	
Radio Signal	X	X	
Radio Transmitter	X	X	
Remove Entity	X	X	
Repair Complete	X	X	
Repair Response	X	X	
Resupply Cancel	X	X	
Resupply Offer	X	X	
Resupply Received	X	X	
Service Request	X	X	
Set Data	X	X	
Start	X	X	
Stop	X	X	
Transfer Control	X	X	

1.4. Translating Between DIS and the RPR FOM

DIS exercises send updates at regular intervals, called heartbeats. HLA does not have this feature. VR-Exchange handles this difference as follows:

- VR-Exchange provides heartbeat updates for the appropriate PDUs when the time between HLA updates exceeds the heartbeat threshold.
- VR-Exchange deletes HLA objects when the time between DIS PDUs exceeds the timeout threshold (2.5*heartbeat). RPR FOM embedded system objects (radio transmitter/receiver and emitter system/beams) are deleted only if their host entity no longer exists.

VR-Exchange does not support the full set of PDUs in the 1278.1a specification. It supports the following PDUs (listed by protocol families):

- Entity Information and Entity Interaction PDUs
  - Entity State PDU
  - Collision PDU
- Warfare PDUs
  - Fire PDU
  - Detonation PDU
- Simulation Management PDUs
  - Start/Resume PDU
  - Stop/Freeze PDU
  - Acknowledge PDU
  - Action Request PDU
  - Action Response PDU
  - Data Query PDU
  - Set Data PDU
  - Data PDU
  - Event Report PDU
  - Comment PDU
  - Create Entity PDU
  - Remove Entity PDU
- Radio Communications PDUs
  - Transmitter PDU
  - Signal PDU
  - Receiver PDU
- Distributed Emission Regeneration PDUs
  - Electromagnetic Emission PDU
  - Designator PDU
  - IFF PDU
- Logistics PDUs
  - Service Request PDU
  - Resupply Offer PDU
  - Resupply Received PDU
  - Resupply Cancel PDU
  - Repair Complete PDU
  - Repair Response PDU
- Entity Management PDUs
  - Transfer Control Request PDU

- Synthetic Environment PDUs
  - Areal Object PDU
  - Environmental Process PDU
  - Gridded Data PDU
  - Linear Object PDU.
- Logger Control PDUs (for MAK Logger)

The following sections describe the supported PDUs in greater detail.

1.4.1. Entity Information and Entity Interaction

The DIS Entity Information/Interaction PDUs are the Entity State PDU and the Collision PDU.

The Entity State PDU is translated into a RPR FOM object. The RPR FOM has broken down the Entity State PDU into an object class hierarchy rooted at the *BaseEntity* class with several levels of subclasses as shown in Table 1-2. The object class that is registered in the federation execution is dependent on the value of the entity type field. The default mapping between the entity type field and the RPR FOM object class is based on the entity type kind and domain values. For example, a DIS M1 tank would be instantiated as a *GroundVehicle* object in the RPR FOM, whereas a TOW missile would be a *MunitionEntity*.

The data fields of the Entity State PDU are translated to and from the corresponding attributes of the appropriate RPR FOM object class. The entity ID, entity type, and position attributes are required to send an Entity State PDU.

Table 1-2: BaseEntity Object Classes

Class1	Class2	Class3	Class4
BaseEntity	PhysicalEntity	PlatformEntity	Aircraft
			AmphibiousVehicle
			GroundVehicle
			MultiDomainPlatform
			Spacecraft
			SubmersibleVehicle
			SurfaceVessel
		Lifeform	Human
			NonHuman
		CulturalFeature	
		Expendables	
		Munition	
		Radio	
		Sensor	

The Collision PDU is translated into a RPR FOM *Collision* interaction. The *Collision* interaction is defined at a single level, with no inheritance.

1.4.2. Warfare

The DIS Warfare PDUs are the Fire PDU and the Detonation PDU, which are translated into RPR FOM *WeaponFire* and *MunitionDetonation* interactions, respectively. Each interaction class is defined at a single level, with no inheritance.

1.4.3. Simulation Management

The DIS Simulation Management (SIMAN) PDUs are translated into the corresponding RPR FOM interactions as shown in Table 1-3.

Table 1-3: SIMAN Representation in the RPR FOM

DIS PDU	RPR FOM Interaction Class
Create Entity	CreateEntity
Remove Entity	RemoveEntity
Acknowledge	Acknowledge
Set Data	SetData
Data Query	DataQuery
Data	Data
Action Request	ActionRequest
Action Response	ActionResponse
Event Report	EventReport
Comment	Comment
Start/Resume	StartResume
Stop/Freeze	StopFreeze

1.4.4. Radio Communications

The DIS Radio Communications PDUs are the Transmitter, Receiver, and Signal PDUs. The Receiver and Transmitter PDUs are translated into RPR FOM objects as *RadioReceiver* and *RadioTransmitter* object classes, respectively. The RPR FOM has broken down the Radio Transmitter and Receiver protocols into an object class hierarchy as shown in Table 1-4. The *EmbeddedSystem* root object class captures the common attributes that specify the objects as being attached to other objects. The specific object class that is used to register a radio object in the federation execution is dependent on the type of PDU received by VR-Exchange. The data fields of the Transmitter and Receiver protocol are translated to and from the corresponding attributes of the appropriate RPR FOM object class. The host entity ID or host object ID (mapping to an entity ID) is required to send a Radio Transmitter PDU.

Table 1-4: EmbeddedSystem Object Classes

Class1	Class2	Class3
EmbeddedSystem	Designator	
	EmitterSystem	
	RadioReceiver	
	RadioTransmitter	
	IFF	IffInterrogator
		IffTransponder

The Signal protocol is translated into one of several RPR FOM *RadioSignal* interaction classes. A base RadioSignal interaction is defined from which the *ApplicationSpecificRadioSignal*, *DatabaseIndexRadioSignal*, *EncodedAudioRadioSignal*, and *RawBinaryRadioSignal* are derived. The HLA signal interaction that VR-Exchange generates is based on the Encoding Class in the DIS Signal PDU.

Table 1-5: RadioSignal Interaction Class

Class1	Class2
RadioSignal	ApplicationSpecificRadioSignal
	DatabaseIndexRadioSignal
	EncodedAudioRadioSignal
	RawBinaryRadioSignal

1.4.5. Distributed Emission Regeneration

The Distributed Emission Regeneration protocol family consists of the Electromagnetic Emission (EE), Designator, and IFF protocols. The EE PDU is transformed into several RPR FOM object classes. The Designator and IFF PDUs are represented in the RPR FOM by a derivation of the *EmbeddedSystem* class as Designator and IFF respectively (see Table 1-4). The IFF object class is further broken down into the *IffTransponder* and *IffInterrogator* object classes. These classes allow subscription based filtering of IFF data.

The EE PDU translation is different from any of the other PDU transformations in that it is a many-to-one and one-to-many transformation. A separate RPR FOM object is created for each emitter system and emitter beam in an EE PDU. Conversely, the emitter system and emitter beam objects reflected to VR-Exchange must be combined into a single EE PDU (for the State Update). The emitter system data is transformed into an *EmitterSystem* object class. An *EmitterSystem* object is created for each emitter system record in the EE PDU. The *EmitterSystem* is derived from the *EmbeddedSystem* class that associates each EmitterSystem object with its host entity object. Based on the beam function, the EE PDU emitter beam data is transformed into either a *RadarBeam* (the default) or a *JammerBeam*. The *EmitterBeam* class contains all of the emitter beam data parameters. Similar to the *EmbeddedSystem*, the *EmitterBeam* defines an *EmittingSystemID* that establishes the association between it and its EmitterSystem object instance.

VR-Exchange creates a Delta State EE PDU for each *EmitterSystem* or *EmitterBeam* reflection. VR-Exchange does not generate a delta EE PDU if the *EmitterSystem* has no active beams (unless that beam is being removed). VR-Exchange must also first receive the *EmitterBeam's EmitterSystem* data as well as the emitting entity data. The emitting system ID and beam ID are required for sending an EE PDU. VR-Exchange generates the State Update EE PDU at a heartbeat rate (regardless of intermediate delta updates). The State Update EE PDU contains the emission data for each emitter with active emitter beams.

1.4.6. Logistics

The DIS Logistics PDUs are translated into RPR FOM interaction classes as shown in Table 1-6.

Table 1-6: Logistics Representation in the RPR FOM

DIS PDU	RPR FOM Interaction Class
Service Request	ServiceRequest
Resupply Offer	ResupplyOffer
Resupply Received	ResupplyReceived
Resupply Cancel	ResupplyCancel
Repair Complete	RepairComplete
Repair Response	RepairResponse

The object ID attributes (mappable to DIS IDs) for the participating entities are required to send these PDUs.

1.4.7. Entity Management

The Transfer Control Request PDU is translated to the HLA *TransferControl* interaction class. The *TransferControl* class is available only if you are using RPR FOM version 2, draft 6 or later.

1.4.8. Synthetic Environment

Table 1-7 lists the PDUs in the Synthetic Environment family that VR-Exchange supports.

Table 1-7: Synthetic environment PDUs

DIS PDU	RPR FOM Class
Areal Object	*ArealObject*
Environmental Process	*EnvironmentProcess*
Gridded Data	*GriddedData*
Linear Object	*LinearObject*

1.4.9. MAK Logger Control PDUs

In addition to the standard DIS PDUs described in previous sections, VR-Exchange supports Logger Control PDUs for controlling the MAK Logger.

1.5. Hardware Requirements and Product Limitations

VR-Exchange was designed to work with up to 254 connections. However, the requirement that all brokers run on the same machine limits the actual number of connections that you can run. VR-Exchange has been tested with as many as 4 connections.

It is recommended that VR-Exchange be run with one processor for each connection. Though this is not required, and much testing has been done on single processor machines, a typical instance of VR-Exchange will have three or more processes competing for CPU time.

It is also recommended that each protocol supported exist on a dedicated network. This is not a requirement of VR-Exchange, but is recommended for high traffic exercises.

1.6. The Limitations of Translation Software

Translating between two protocols, or even two FOMs (in the case of HLA) is similar to translating between two spoken languages, in that you may lose something in translation. The goal of a good translator is to try to minimize that loss. When translating between spoken languages, you cannot simply perform word for word replacement. The source text needs to be examined holistically, and translated so that it conveys the same meaning to the target language. Sometimes idiomatic structures are not directly translatable and can only be approximated in the target language. And some concepts present in one language may not even exist in the other. As a result, some precision may be lost.

This same limitation occurs with all software translators. Messages often cannot be translated on a packet by packet basis; rather a stream of messages as a whole must be examined and translated. Some protocols map easily to each other. Others do not. Some protocols have no valid translation at all.

In the case of DIS to HLA translation, or even FOM-to-FOM translation, there is not always a one-to-one mapping between PDUs received from the DIS side and updates sent to the HLA side (or vice versa). This is due to fundamental differences in the rules and conventions of HLA versus DIS, and to differences in the ways different FOMs choose to represent similar concepts. For example, the DIS broker needs to send DIS heartbeat PDUs every 5 seconds, even though it may not receive HLA updates for an object for a much longer period of time. A single DIS PDU might contain data that must be translated into a series of attribute updates. For example, in DIS, a single Electromagnetic Emission PDU might contain information about a series of emitter systems and beams, but in HLA's RPR FOM, each emitter system and beam exists as a separate object, and must be updated through separate streams of messages.

In general, the strategy that VR-Exchange uses is to decode incoming data into state repositories that always contain the current state of each object. State repository information is passed to the other side of the bridge each frame, independently of when actual updates might have arrived. Then, the publishing rules of the other protocol and/or FOM are used to determine when to send out data describing the object. A consequence of this strategy is that no one piece of code is ever looking at both an incoming message and an outgoing message at the same time. While this strategy is necessary to support FOM agility and generic protocol-translation in VR-Exchange, a limitation is that “something” can get lost in the translation. For example, you should not expect that the timestamp on an outgoing HLA update will match the timestamp of a particular incoming DIS PDU.

Translators are very valuable solutions to real-world interoperabilty problems, and they often offer a much cheaper and easier solution than re-tooling applications to use a common protocol. But they can never quite provide the same level of close-coupling that you can get if all of your systems communicated natively. MAK works hard to insure that our translators retain as much of the meaning and semantics of the data as is feasible, given the fundamental differences in the various protocols. But users must be aware that, as with any type of translator, there will always be limitations.

2. Installing VR-Exchange

This chapter explains how to install VR-Exchange and associated software.

2.1. Installing VR-Exchange

A full VR-Exchange installation includes the application, the License Manager software, and for HLA operation, an RTI (or Runtime Infrastructure.)

To install the toolkit, you must enter an installation code. Be sure you have received your code from your MAK salesperson before you start the installation process. If you do not have an installation code, you can install the runtime files. When you receive your installation code you can install the toolkit.

Before you install VR-Exchange, please read the Release Notes to see if there are any special instructions for installation.

2.1.1. Installing VR-Exchange on Windows

When you install large applications on Windows, there may be a delay of up to several minutes from the time you try to run the setup program to the time that an installation dialog box is displayed. This is due to how Windows scans setup programs before it executes them. If you experience this problem, turning off User Access Control can reduce or eliminate this delay.

Windows versions of VR-Exchange are provided as executable installer files on DVD, or as downloaded files. The installers are named to indicate the compiler used to build that version of VR-Exchange

➤ To install VR-Exchange, run the installer. Follow the instructions in the installation wizard.

2.1.2. Installing VR-Exchange on a Linux System

Linux versions of VR-Exchange are provided as compressed tar files on DVD, or as downloaded files.

To install VR-Exchange on Linux:

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

   `tar -vxzf application.tar.gz`

   where *application* is the product release identification.

When you uninstall , the uninstaller does not uninstall the data package. You must delete the files manually.

2.2. Installing and Setting Up the MAK License Manager

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

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

The License Manager installer is included on MAK installation media. It is separate from the product installers. You can download the installers from our web site at https://www.mak.com/support/license-support or you can download directly from:

Windows (32 bit): http://ftp.mak.com/out/MAKLicenseManager-win-setup.exe

Windows (64 bit): http://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe

Linux (32 bit): http://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz

Linux (64 bit): http://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

Complete installation and configuration instructions are included with the License Manager installer. Instructions are also available at the license support page.

Some customers use dongle licenses instead of running a license server. Instructions for using dongles are in the License Manager documentation.

2.2.1. Specifying the License Server

The first time you run a MAK application on a particular computer, the License Setup dialog box opens (Figure 2-1). It prompts you to enter the hostname of the license server and optionally, a port number.

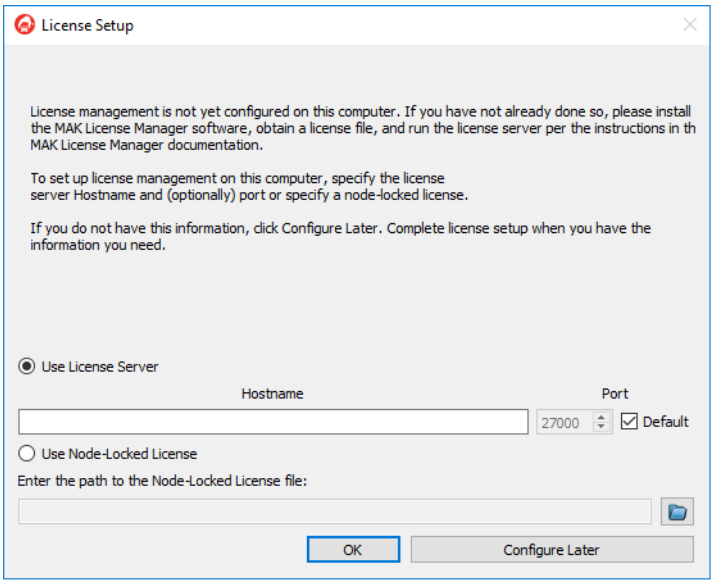

Figure 2-1. License Setup dialog box

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

If you know the hostname, type it in the Hostname box. Then click OK. The application will start.

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

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

The MAKLMGRD_LICENSE_FILE Environment Variable

As an alternative to using the License Setup procedure described in the previous section, you can configure the license server in an environment variable. The MAKLMGRD_LICENSE_FILE environment variable identifies the server machine. If you set this environment variable, it overrides the settings stored by the License Setup procedure.

The syntax for the environment variable is: `@Server_name`. For example, if the server machine is oak, set the environment variable to `@oak`.

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

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

1. On the Start menu, click the Settings button. The Settings window opens.
2. In the search box type environment. A list of choices is displayed.
3. Select Edit the System Environment Variables. The System Properties dialog box opens.
4. Click Environment Variables. The Environment Variables dialog box opens.
5. In the System Variables group box, click New. The New System Variable dialog box opens.
6. In the Variable Name field, enter MAKLMGRD_LICENSE_FILE.
7. In the Variable Value field, enter `@server_name`, where `server_name` is the name of the license server.
8. Click OK to back out of each dialog box and set the variable.

Linux

On Linux, you set environment variables in your *.cshrc* (or equivalent startup file). Set the variable similarly to the following example:

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

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

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

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

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

2.3. Installing an RTI

An RTI is a software library (and perhaps supporting executables) that implements an HLA Interface Specification. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications need to use an RTI.

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

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

For the most recent information about the RTI versions supported by MAK products, please see the release notes for your MAK application.

2.3.1. Installing the MAK RTI

To install the MAK RTI, follow the instructions in Chapter 2 of *MAK RTI Users Guide*.

Configuring Your System to Use the MAK RTI

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

- Linux — LD_LIBRARY_PATH.
- Windows — PATH.

The MAK RTI needs to know where to find the following configuration files:

- The federation configuration file (*.fed*, *.fdd*, or *.xml*) for your federation execution (required).
- The RID file (*rid.mtl*) (optional).

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

Running Applications with the MAK RTI

To run a MAK application with the MAK RTI:

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

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

2.4. Installing DDS Software

The Object Management Group (OMG) Data Distribution Service (DDS) is a standard for exchanging messages between applications. VR-Exchange includes two DDS brokers, the OpenSplice DDS broker and the RTI DDS broker.

The VC14 version of VR-Exchange only supports the RTI DDS broker.

2.4.1. The OpenSplice DDS Broker

The OpenSplice DDS broker is built using the OpenSplice DDS Community Edition implementation of OMG-DDS. If you want to run or build the DDS broker, you must install OpenSplice DDS Community Edition. You can be download it from http://www.prismtech.com/dds-community. Please check VR-Exchange *2.6 Release Notes* for the version of DDS supported.

Install OpenSplice DDS according to the instructions provided with the software. This may just require unzipping and untarring the installation package. On Windows, you may just need to run an installer executable.

After you install the DDS software, you must set environment variables. On Windows, you can set the environment variables by running release.bat. On Linux, you can run release.com. However, running one of these scripts just sets the environment for the command shell in which it is run. Therefore, it is recommended that you set the environment globally. The variables that you must set are:

- SPLICE_ORB=DDS_OpenFusion_1_6_1
- OSPL_HOME=*path to your OpenSplice DDS directory*
- OSPL_TARGET=x86_64.win64 (or the appropriate setting for your platform)
- PATH=%OSPL_HOME%\bin;%OSPL_HOME%\lib;%OSPL_HOME%\examples\lib;%PATH%
- OSPL_TMPL_PATH=%OSPL_HOME%\etc\idlpp
- OSPL_URI="file://%OSPL_HOME%\etc\config\ospl.xml"

2.4.2. The RTI Connext DDS Broker

The RTI Connext DDS broker is built using RTI Connext DDS. If you want to run or build the RTI DDS broker, you must install the RTI Connext DDS package. You can download it from https://www.rti.com/.

Install RTI Connext DDS according to the instructions provided with the software. This may require running an installer executable on Windows or extracting a package on Linux.

After you install the software, set the NDDSHOME environment variable to the path to your RTI Connext DDS directory.

Add the NDDSHOME variable to your path:

```
PATH=%NDDSHOME%/lib/COMPILER
```

where COMPILER refers to the subdirectory under *lib* that is specific to different compilers.

3. Using VR-Exchange

This chapter explains how to start the Portal and use it to manage connections.

3.1. Starting VR-Exchange

This section explains how to start VR-Exchange on the different operating systems it supports.

All VR-Exchange components (the Portal and brokers) must run on the same computer.

3.1.1. Starting VR-Exchange on Linux

To start VR-Exchange on Linux:

1. In a command shell, change directory to *./bin*.
2. Enter:

   `vrx [command-line_options]`

The VR-Exchange Portal window opens (Figure 3-1). The VR-Exchange command-line options are described in "VR-Exchange Command-Line Options," on page 3-4.

3.1.2. Starting VR-Exchange on Windows

➤ To start VR-Exchange on Windows, on the Start menu, choose **All Programs** → **MAK Technologies** → **VR-Exchange 2.6** → **VR-Exchange (32-bit)** (or 64-bit).

The VR-Exchange Portal window opens (Figure 3-1).

If you want to start VR-Exchange with command-line options, start it from a command prompt or create a shortcut icon for it on your desktop. The VR-Exchange command-line options are described in "VR-Exchange Command-Line Options," on page 3-4.

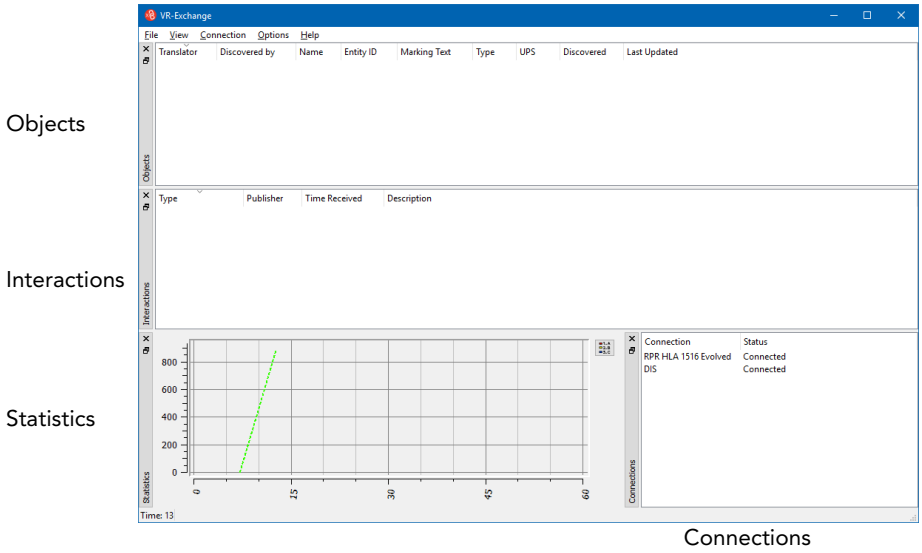

Figure 3-1. VR-Exchange Portal window

3.1.3. VR-Exchange Command-Line Options

The VR-Exchange command-line options override settings in *VR-Exchange.xml*. Table 3-1 describes the command-line options

Table 3-1: VR-Exchange Portal command-line options

Option	Description
`(-- | --ignore_rest)`	Tells VR-Exchange to ignore the arguments following this flag.
`--controlQueueName` *queuename*	Specifies the name of the control shared memory queue.
`--dontStartBrokers`	Specifies that VR-Exchange not enable the connections listed in *VR-Exchange.xml*. If you use this option, you must start the connections manually.
`(-G | --noGui)`	Specifies console mode (no GUI).
`(-h | --help)`	Displays usage information and exits.
`--interaction-QueueName` *queuename*	Specifies the name of the interaction shared memory queue.
`(-l | --configFile)` *filename*	Specifies a configuration file other than *VR-Exchange.xml*.
`(-L | --logFilename)` *filename*	Specifies a log file.
`--maxInteractionsTo-Show` *count*	Specifies the maximum number of interactions to show in the Portal application.
`--maxPortalControlMes-sages` *count*	Specifies the maximum number of control messages to process per tick.
`--maxPortalInterac-tionMessages` *count*	Specifies the maximum number of interaction messages to process per tick.
`--maxPortalObjectMes-sages` *count*	Specifies the maximum number of object messages to process per tick.
`(-n | --notifyLevel)` *notify_level*	Specifies the notification level for logging messages. Range: 0-4, where: ◆ 0 – Fatal error messages. ◆ 1 – Warning messages. ◆ 2 – Informational messages. ◆ 3 – Verbose information. ◆ 4 – Debugging data.
`--objectQueueName` *queuename*	Specifies the name of the object shared memory queue.
`--queueBucketCount` *count*	Specifies the number of buckets inside the shared memory queue.
`--queueBucketPayload-Size` *size*	Specifies the size of buckets inside the shared memory queue.

Table 3-1: VR-Exchange Portal command-line options

Option	Description
`--tcpPort` *`port_number`*	Specifies the TCP port for connectivity tests.
`(-v | --version)`	Displays version information and exits.

3.1.4. The Portal Application

The Portal application houses several dockable windows. You can drag any of them off of the Portal and place them anyplace on your desktop. You can close any of the windows and open them to suit your viewing preferences.

The Portal has context-sensitive menus. The windows have a context-sensitive menu that matches the View menu. The individual connections in the Connections window have context-sensitive menus that match the Connection menu.

3.1.5. Running VR-Exchange without the Portal Application

You can run VR-Exchange without the Portal application. It automatically starts any connections that are configured to start when the Portal starts.

To run VR-Exchange without the Portal application:

1. Open *./bin/VR-Exchange.xml* in a text editor.
2. Set the value of the `noGui` attribute to true.

```
<!--Disable Portal GUI-->
<noGui value="true"/>
```

3. Save the file.
4. Start VR-Exchange.

3.2. Configuring VR-Exchange

VR-Exchange has several parameters that you can set to optimize performance. Preferences are saved to *VR-Exchange.xml*.

To set VR-Exchange preferences:

1. Choose **Options** → **Preferences**. The VR-Exchange Preferences dialog box opens (Figure 3-2).

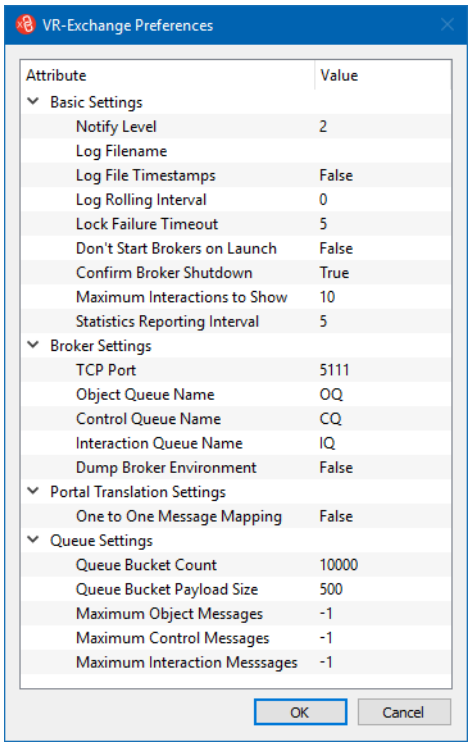

Figure 3-2. VR-Exchange Preferences dialog box

2. Expand the appropriate group of settings.
3. Click the value for the attribute that you want to change. The value becomes editable.
4. Edit the value. Some values can be chosen from a drop-down list. Others are text or numeric values that you edit directly.
5. Click OK. Changes do not take effect until you restart VR-Exchange.

Table 3-2 describes the attributes in the VR-Exchange Preferences dialog box.

Table 3-2: VR-Exchange preferences

Attribute	Description
Basic Settings	
Notify Level	Specifies the output level for the Portal's log file. Default: 2.
Log Filename	Specifies the log file for messages. Default: *disBroker.log*, *rpr13Broker.log*, and so on.
Log File Timestamps	Enable or disable printing timestamps to the log file.
Log Rolling Interval	Frequency, in seconds, to roll the log file to a new file.
Lock Failure Timeout	How long, in seconds, to wait for shared memory locks before retrying.
Don't Start Brokers on Launch	Specifies that connections should not be enabled automatically when VR-Exchange starts.
Confirm Broker Shutdown	Enable or disable prompting for confirmation before shutting down a broker.
Maximum Interactions to Show	Specifies the maximum number of interactions to show in the Interactions window for each interaction type. When the Portal runs for a long time, the interaction list can become very long, which can degrade performance. If this number is set to -1, all interactions are shown. If it is set to any positive number, only the most recent maximum number of interactions is displayed. Default: 10.
Statistics Reporting Interval	The amount of time, in seconds, between updates of the Statistics graph by individual brokers and the Portal.
Broker Communication Settings	
TCP Port	Specifies the TCP port to use to test the broker lifetime. Each broker connects to this port. If the Portal crashes, the brokers will know and then shut down. If TCP Port is set to 0, this feature is disabled.
Object Queue Name Control Queue Name Interaction Queue Name	Specifies the names of the object, control, and interaction queues in the shared memory pool. If you run more than one instance of VR-Exchange, the queue names must be different for each Portal.
Dump Broker Environment	Print the brokers environment to a file at startup.
Portal Translation Settings	
One to One Message Mapping	Specifies whether or not to generate an outbound message for each inbound message. Changing this setting requires restarting all brokers.

Table 3-2: VR-Exchange preferences

Attribute	Description
Queue Settings	
Queue Bucket Count	Specifies the maximum number of messages in the shared memory queue.
Queue Bucket Payload Size	Specifies the maximum message size, in bytes.
Maximum Control Messages Maximum Interaction Messages Maximum Object Messages	Specifies the maximum number of Portal messages to process per call to drainInput(), where -1 means process all messages. Adjust these parameters to avoid process starvation if one of the queues is heavily used.

3.2.1. Additional Portal Configuration Options

The portal has some configuration options that affect performance, but which are not configurable through the GUI. You can configure them in *VR-Exchange.xml*.

- maxObjectViewCount. Specifies the maximum number of objects to display in the portal (per category). When the maximum number of objects is reached, a warning is printed to the log. Default: 5000.
- objectAutocollapseCount. Specifies the number of objects at which to automatically collapse an object category. Default: 1000.

3.3. Enabling and Disabling Connections

The brokers are executable files that are independent of the Portal. You start them using connection configurations that specify a broker, a connection name, and configuration parameters. You can enable a connection automatically when the Portal is started or you can enable a connection manually. For details about adding and configuring connections, please see Chapter 4, *Working with Broker Connections*.

3.3.1. Enabling Connections when the Portal Starts

To specify that a connection be enabled when the Portal starts:

1. Start VR-Exchange.
2. In the Connections window (Figure 3-1), select the connection that you want to configure.
3. If the connection is enabled, choose **Connection** → **Disable Connection**. The connection is disabled.
4. Choose **Connection** → **Edit Connection**. The Edit Connection dialog box opens (Figure 3-3).

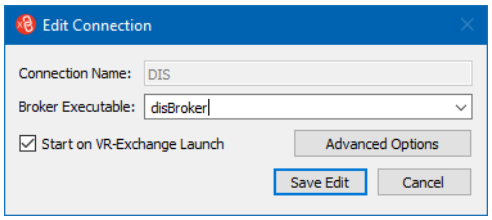

Figure 3-3. Edit Connection dialog box

5. Select the Start on VR-Exchange Launch check box.
6. Click Save Edit.
7. Enable the connection, if applicable.

3.3.2. Enabling Connections During Runtime

To enable a connection after you start the Portal:

1. In the Connections window (Figure 3-1), select the connection that you want to enable.
2. Choose **Connection** → **Enable Connection**. The status changes to Connected.

3.3.3. Disabling Connections

Connections are disabled and brokers are shut down automatically when the Portal is shut down. Connections can also be disabled independently. If a broker crashes, or otherwise exits, and if the broker is connected using TCP, the Portal removes any connections that use it.

To disable a connection:

1. In the Connections window (Figure 3-1), select the connection you want to disconnect.
2. Choose **Connection** → **Disable Connection**.

3.4. Viewing Objects and Interactions

VR-Exchange displays lists of the objects and interactions that it discovers, which connection is publishing them, and statistics about them (Figure 3-4). As objects join and leave the exercise, VR-Exchange updates the display so that it only shows currently simulated objects. You can configure the number of interactions kept in the interaction list. (For details, please see "Configuring VR-Exchange," on page 3-6.)

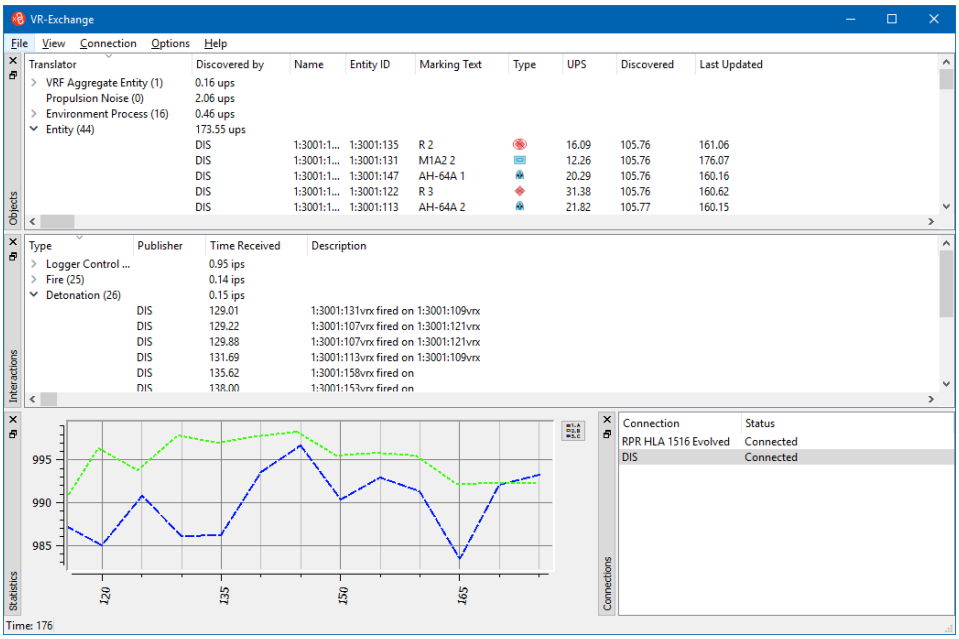

Figure 3-4. Object and interaction display

If you mouse over the icon for an entity in the Type column, VR-Exchange displays a tooltip with the entity type enumeration for that object.

3.4.1. Customizing the Display of Object and Interaction Data

You can choose which columns of data to display and you can change the order in which columns are displayed.

To show or hide a column:

1. In the Objects window or Interactions window (Figure 3-4), right-click any column label. A list of columns is displayed.
2. To display a hidden column or hide a currently displayed column, click its entry in the list.

➤ To reorganize the order of data columns, drag the column title to the location where you want to display it.

3.4.2. Enabling and Disabling the Display of Object Data

You can enable and disable display of the following types of object data:

- Objects – show or hide all objects
- Updates – show or hide the Updated column.

➤ To enable or disable display of object data, choose **View** → **Objects** → *option*, where *option* is one of the following:

– Objects

– Updates.

3.4.3. Sorting the Object List by Column

➤ To sort the object list by column, click a column heading.

3.4.4. Searching the Object List

When there are many objects listed in the Objects window, it can be difficult to find a particular object that you want to examine. You can search the objects list to quickly find particular objects. VR-Exchange searches the Name, Entity ID, and Marking Text columns in sequence to find the search string.

To search the Objects list:

1. Choose **View** → **Objects** → **Find Objects** or click in the Objects window and press **`CTRL+F`**. The Find dialog box opens.
2. Type the string that you want to search for. As you type, VR-Exchange highlights objects that match the search string (Figure 3-5).

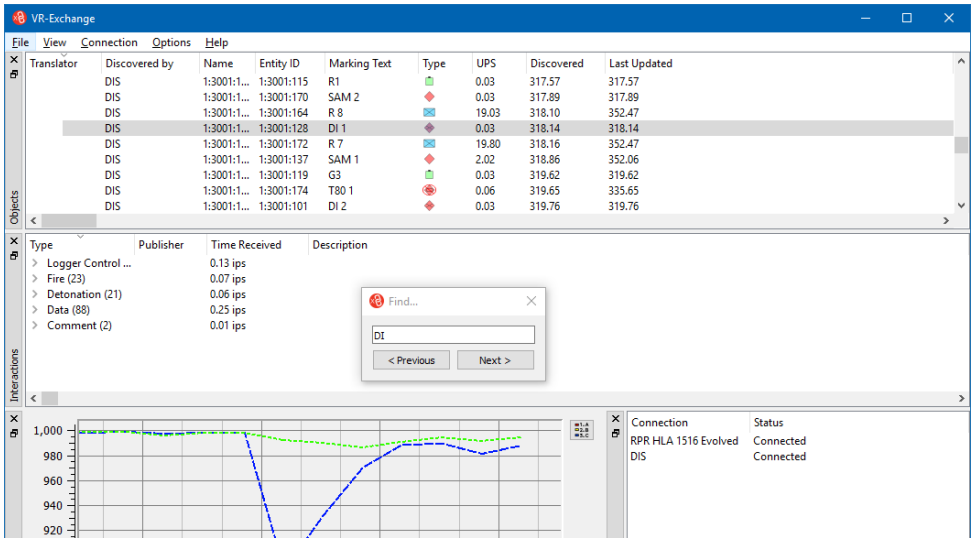

Figure 3-5. Searching the objects list

3. Press **`Enter`**, or click the Next> button on the Find dialog box to cycle forward through the list of matching objects. Click <Previous to iterate backwards through matching objects.

3.4.5. Enabling and Disabling the Display of Interactions

You can hide or display the interaction list.

➤ To enable or disable the display of interactions, choose **View** → **Interactions** → **Interactions**.

3.4.6. Clearing the Interactions List

You can clear the Interactions list. When you clear the interactions list, VR-Exchange continues to maintain a running count of the number of interactions. Figure 3-6 illustrates the interactions list after it is cleared. Notice the cumulative statistics next to Fire and Detonation.

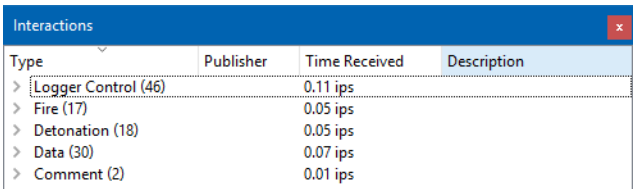

Figure 3-6. Cleared Interactions list

➤ To clear the Interactions list, choose **View** → **Interactions** → **Clear Interactions**.

Resetting Statistics

You can clear the cumulative statistics maintained by the Interactions window.

➤ To clear interaction statistics, choose **View** → **Interactions** → **Reset Statistics**.

3.4.7. Expanding and Collapsing the Tree View

You can quickly expand and collapse all entries in the the Objects and Interactions windows.

---

i A category automatically collapses if the number of entries exceeds the value specified by the **objectAutocollapseCount** parameter in *VR-Exchange.xml*. Default: 1000.

To expand and collapse the tree view:

1. In the Objects window or Interactions window, right-click any entry. A context-sensitive menu is displayed.
2. Choose **Expand All** or **Collapse All**.

3.5. Filtering Objects in the Objects Window

You can filter out entire classes of objects (for details, please see "Specifying Which Translators to Use," on page 4-26). You can also filter individual objects. You might do this if you are interested in particular objects and want to reduce network traffic by filtering out the objects you are not interested in. You can filter objects by name, entity type, and marking text. Filtering by entity type and marking text only applies to entities and aggregates.

This section explains how to filter out objects from the Objects window. You can also set up filters in the Connection Information dialog box for a broker. For details, please see "Specifying Filters in the Connection Information Dialog Box," on page 4-22.

To filter out an object:

1. In the Objects window, select the objects that you want to filter.
2. Right-click the objects. A context-sensitive menu is displayed.
3. Choose **Add Filter**. A submenu is displayed.
4. Choose the type of filter that you want to add. The word "filtered" is displayed in the Translator column for each filtered object (Figure 3-7).

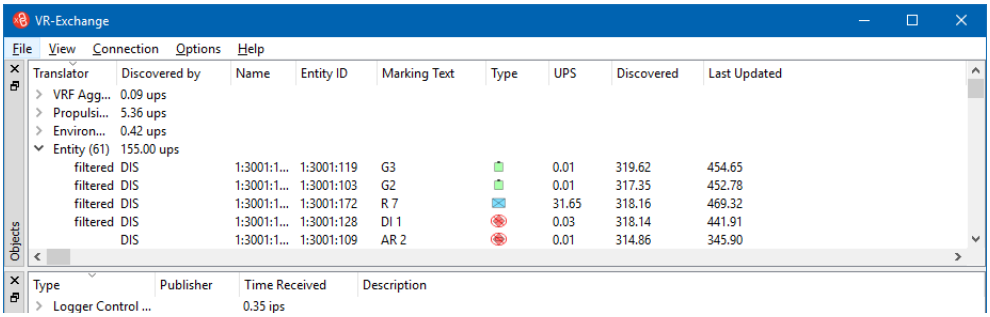

Figure 3-7. Filtered objects

To remove a filter:

1. In the Objects window, select the objects that you want to unfilter.
2. Right-click the objects. A context-sensitive menu is displayed.
3. Choose **Remove Filter**. A submenu is displayed.
4. Choose the type of filter that you want to remove. The filter is removed and the "filtered" text in the Translator column is cleared.

Only filters created from the contest-sensitive menu can be removed from the context-sensitive menu. Filters created on the Filters page of the Connection Information dialog box cannot be removed from the context-sensitive menu.

3.6. Displaying the Queue Status

You can display details about the shared memory queue.

➤ To display the queue status, choose **View** → **Queue Status**. The Queue Status dialog box opens (Figure 3-8).

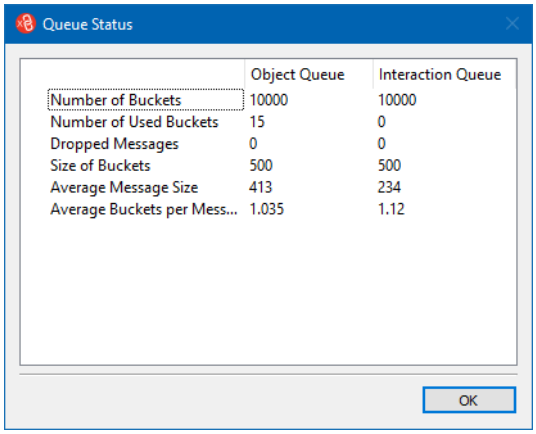

Figure 3-8. Queue Status dialog box

3.7. The Statistics View

You can display a graph of the number of packets being processed by VR-Exchange and internal VR-Exchange messages (Figure 3-9).

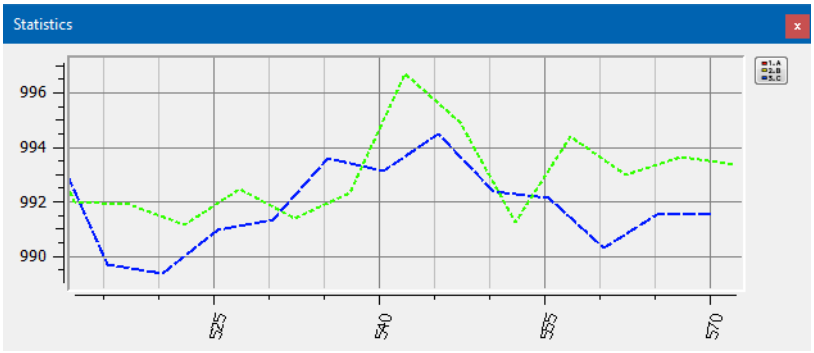

Figure 3-9. Statistics View

➤ To hide or display the Statistics View, choose **View** → **Statistics**.

3.7.1. Displaying the Statistics Legend

The graphs in the Statistics View are color-coded. The Statistics Legend explains the meaning of each graph for each broker (Figure 3-10).

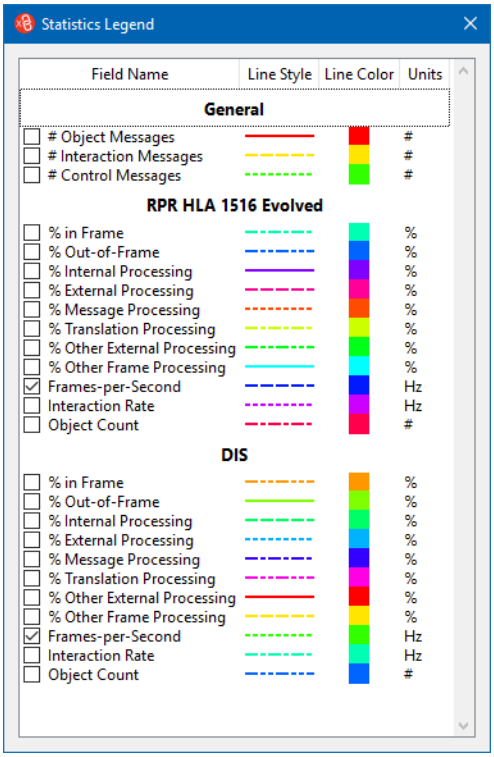

Figure 3-10. Statistics Plot Legend

➤ To display the Statistics View Legend, choose **View** → **Statistics Legend**, or click the Legend icon ( ) on the Statistics View.

3.8. Monitoring Translations

You can monitor translation on a per-object basis. When you monitor an object, VR-Exchange shows the most recent input and output for the object. This information can be useful for debugging brokers.

To monitor translation of an object:

1. In the Objects window, right-click the object you want to monitor. A menu is displayed.
2. Choose **Monitor Translation**. The Translation Monitoring window opens (Figure 3-11).

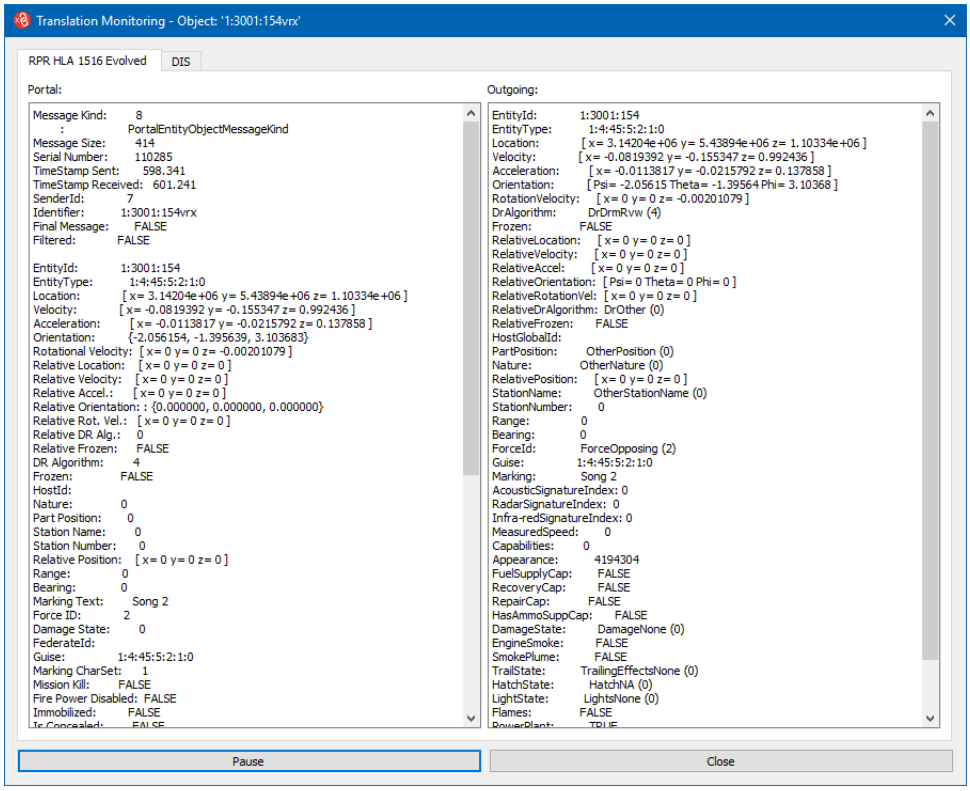

Figure 3-11. Translation Monitoring window

3.9. Closing VR-Exchange

➤ To close VR-Exchange, choose **File** → **Exit**.

3.10. Using vrxMessageDump

VR-Exchange includes a utility called *vrxMessageDump*. It displays all messages passing through shared memory in a console window (Figure 3-12). vrxMessageDump exits automatically when the Portal is shut down.

➤ To run *vrxMessageDump*, start VR-Exchange, then run the *vrxMessage-Dump* executable in the *./bin* directory.

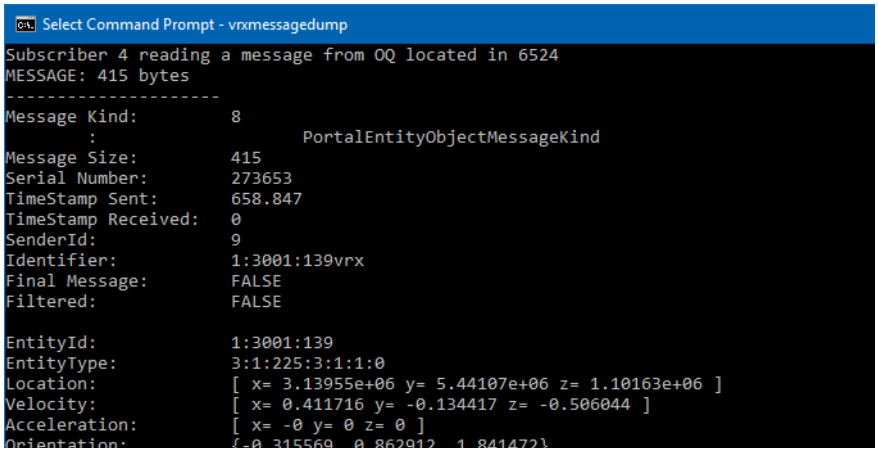

Figure 3-12. vrxMessageDump display

4. Working with Broker Connections

This chapter explains how to add, edit, and configure connections.

4.1. Adding and Editing Connections

A connection runs a broker executable with a unique set of parameters. This allows you to, for example, run an instance of the HLA 1.3 broker with RTI A and another instance of the HLA 1.3 broker with RTI B.

A connection must have a name and it must specify the broker that it uses and a configuration file. It can include command-line options and can specify environment variables. For lists of command-line options, please see Appendix A, *Broker Details*.

VR-Exchange ships with connections for HLA 1.3, DIS, and DDS. The HLA and DIS connections are started automatically when you start VR-Exchange.

4.1.1. Adding Connections

To add a connection:

1. Choose **Connection** → **Add New Connection**. The Add New Connection dialog box opens (Figure 4-1).

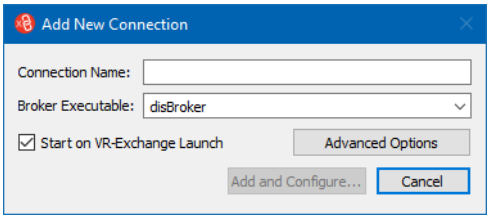

Figure 4-1. Add New Connection dialog box

2. In the Connection Name box, type a name for the connection.
3. In the Broker Executable box, select a broker from the drop-down list or type the name of a broker executable.
4. Optionally, specify advanced options, as follows:
   a. Click Advanced Options. The dialog box expands (Figure 4-2).

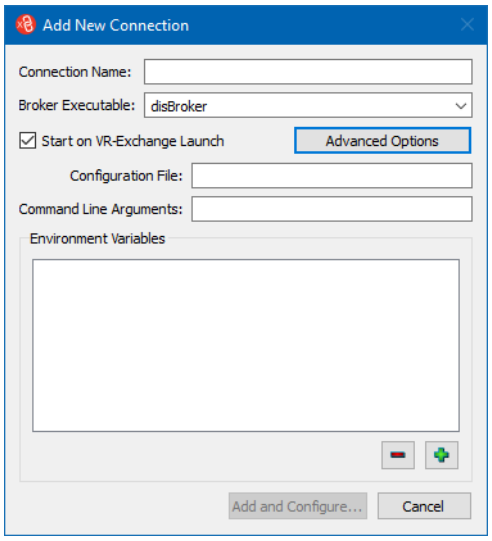

Figure 4-2. Add New Connection dialog box, Advanced Options

b. In the Configuration File box, type a name for the connection configuration file. (VR-Exchange will append the extension *.bcf*.) If you do not specify a configuration file name, VR-Exchange uses the connection name to name the configuration file.

c. Optionally, type command-line arguments in the Command Line Arguments box.

d. Optionally, specify environment variables for this connection, as described in "Specifying Environment Variables for a Connection," on page 4-4.

e. Click Advanced Options to collapse the dialog box.

5. If you want to enable the connection when VR-Exchange starts, select the Start on VR-Exchange Launch check box.

6. Click Add and Configure. The Connection Information dialog box for this connection opens.

7. Optionally, edit configuration parameters for the new connection. For details, please see "Configuring Connections," on page 4-6.

4.1.2. Specifying Environment Variables for a Connection

You can specify environment variables that only apply to a specific connection. You might want to do this to have different HLA connections use different RTIs.

To specify an environment variable for a connection:

1. In the Add New Connection dialog box or Edit Connection dialog box, click Advanced Options. The dialog box expands (Figure 4-2).
2. In the Environment Variables group box, click the Add button (+). The Environment Variable dialog box opens.
3. In the Variable Name box, type a name for the environment variable.
4. In the Variable Value box, type the value for the variable. It can be a fully qualified path or it can use variable expansion, for example:

   `PATH=%PATH%` (on Windows)

   `PATH= .;$PATH` (on Linux)
5. Click OK. The environment variable is added to the list (Figure 4-3).

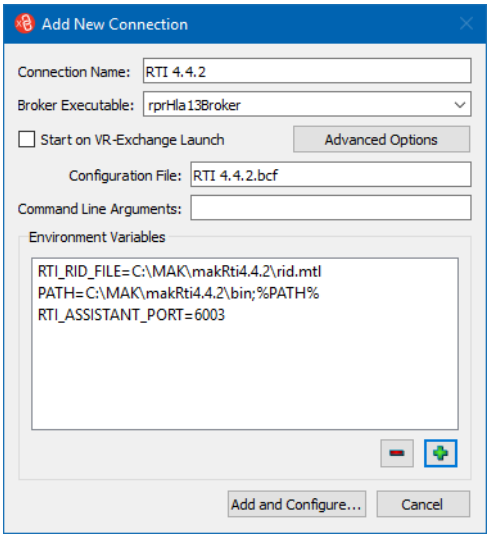

Figure 4-3. Environment variables

4.1.3. Editing Connections

To edit a connection:

1. In the Connections window (Figure 3-1), select the connection that you want to edit.
2. If the connection is connected, choose **Connection** → **Disable Connection**.
3. Choose **Connection** → **Edit Connection**. The Edit Connection dialog box opens (Figure 4-4).

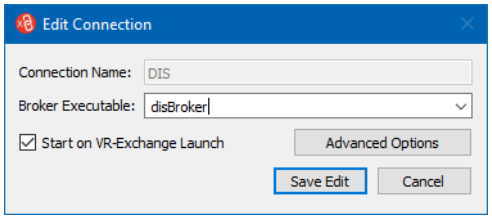

Figure 4-4. Edit Connection dialog box

4. Click Advanced Options to display all connection details.
5. Edit the connection as desired.
6. Click Next.

4.1.4. Deleting Connections

Deleting a connection deletes the association of a connection name with a broker and a configuration file. It does not affect the broker executable referenced by the configuration file and it does not affect the configuration file it references (if one exists).

To delete a connection:

1. In the Connections window (Figure 3-1), select the connection that you want to delete.
2. If the connection is connected, choose **Connection** → **Disable Connection**.
3. Choose **Connection** → **Remove Connection**.
4. Click OK. A confirmation dialog box opens.
5. Click OK.

4.2. Configuring Connections

This section describes the attributes you can configure on the Configure page of the Connection Information dialog box.

Each connection has a set of attributes that you can set. If you use the default attribute values, VR-Exchange does not create a configuration file for the connection. If you change any of the attributes, VR-Exchange saves them to the configuration file you specified when you created the connection.

To configure a connection:

1. In the Connections window (Figure 3-1), select the connection that you want to configure.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens (Figure 4-5). (When you add a new connection, the Connection Information dialog box opens automatically after you create the connection.)

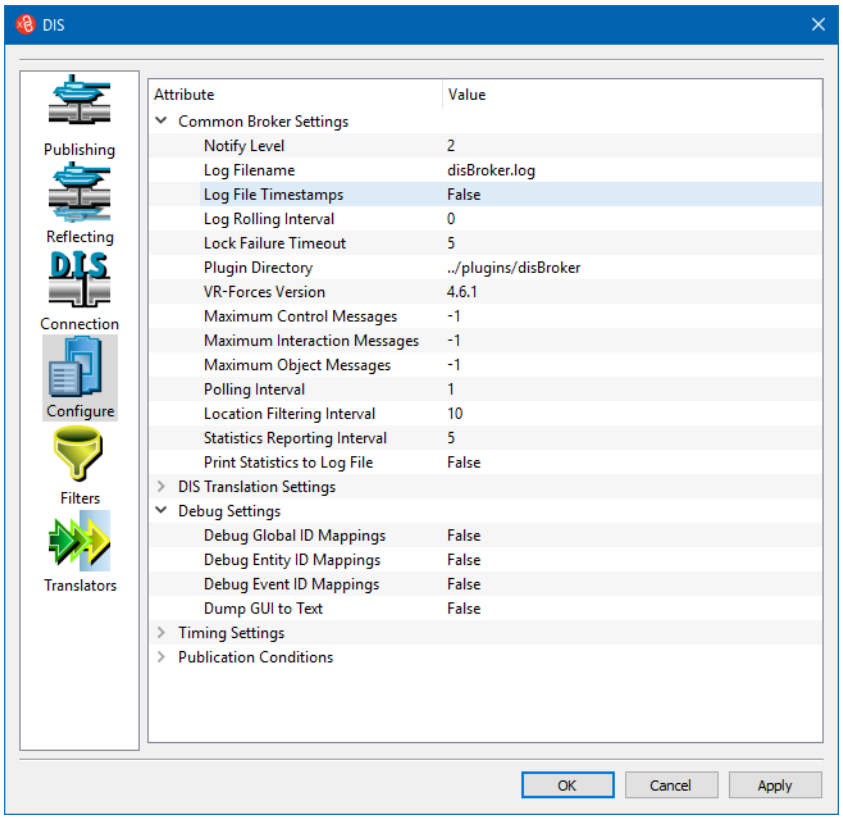

Figure 4-5. Connection Information dialog box

3. Select the Configure page. It lists attributes for the connection, organized in logical groupings, except for the network connection settings. These are configured on the Connections page.
4. Expand the attribute tree to find the attributes you want to edit.

5. To edit an attribute, click its value. The field becomes editable.
   - If the field has a single text or numeric value, change the value as desired. Some values are available on drop-down lists. The attributes are listed and described in later sections in this chapter.
   - If the attribute value is a collection, edit it as described in "Specifying Collection Values," on page 4-7.
6. Click OK. If the connection was enabled, the changes do not take effect until you restart the connection. VR-Exchange prompts you to restart the connection.

4.2.1. Specifying Collection Values

Some attributes can take multiple values (collections).

To specify the values for a collection:

1. In the Configure page for a connection, expand the attribute list to display the collection attribute you want to edit.
2. Click the word Collection in the Value column (Figure 4-6). It changes to a button labeled Click to Edit.

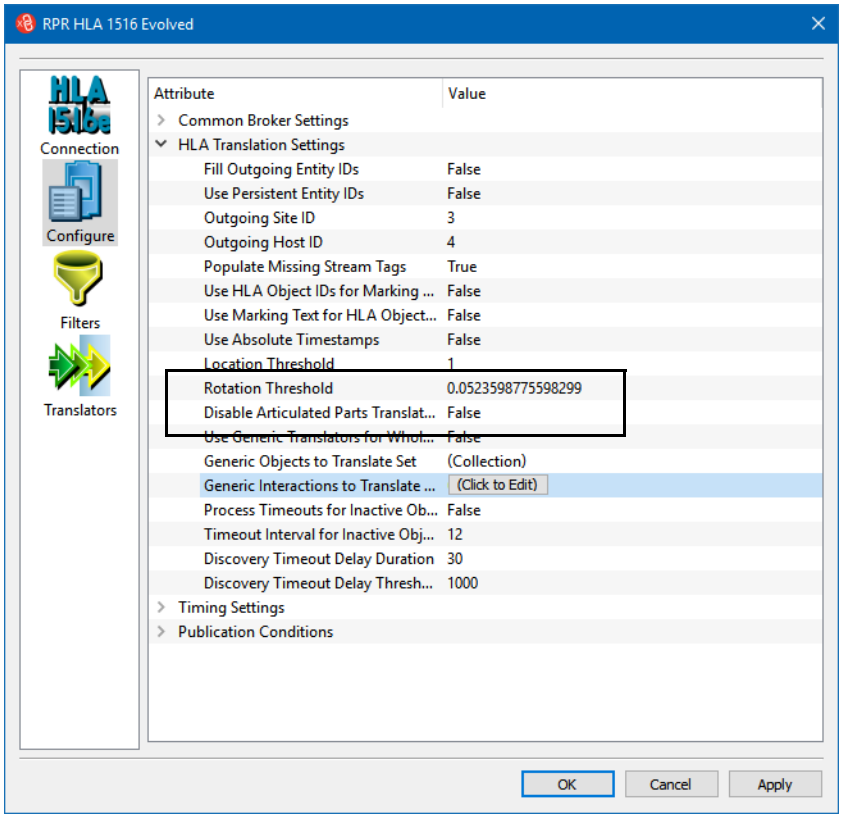

Figure 4-6. Collection value

3. Click the button. The Edit Collection dialog box opens (Figure 4-7).

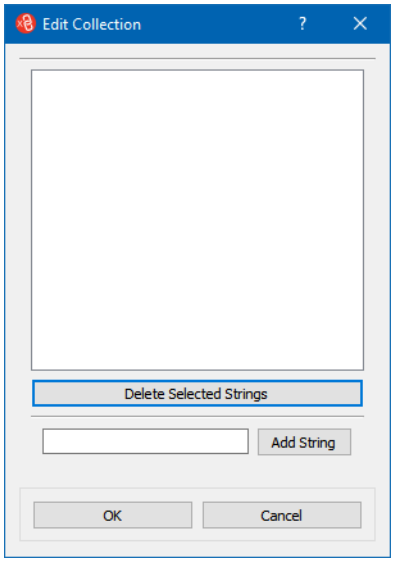

Figure 4-7. Edit Collection dialog box

4. In the text box, type the string you want to add.
5. Click Add String. The string is added to the list window.
6. Click OK.

Removing Values from a Collection

To remove values from a collection:

1. In the Configure page for a connection, click the word Collection in the Value column (Figure 4-6). It changes to the text Click to Edit.
2. Click the button. The Edit Collection dialog box opens (Figure 4-7).
3. In the list window, select the strings that you want to delete.
4. Click Delete Selected Strings.
5. Click OK.

4.2.2. Common Broker Settings

Table 4-1 describes the attributes in the Common Broker Settings group. They are the same for all of the brokers.

Table 4-1: Common broker settings attributes

Attribute	Description
Notify Level	The logging output level for the broker's log file. Range 0-4, where: 0 = Fatal error messages. 1 = Warning messages. 2 = Informational messages. 3 = Verbose information. 4 = Debugging data. Default: 2.
Log Filename	The name of the file to log broker output to. Default: "" (no log file).
Log File Timestamps	Specifies whether or not timestamps are printed to the log file.
Log Rolling Interval	How often, in seconds, to roll the log over to a new file.
Lock Failure Timeout	How long, in seconds, to wait for shared memory locks before retrying.
Plugin Directory	The directory to search in for broker plug-ins.
Maximum Control Messages	The maximum number of Portal control messages to process per call to drainInput(). If set to -1, then all messages are processed. Default: -1.
Maximum Interaction Messages	The maximum number of Portal interaction messages to process per call to drainInput(). If set to -1, then all messages are processed. Default: -1.
Maximum Object Messages	The maximum number of Portal object messages to process per call to drainInput(). If set to -1, then all messages are processed. Default: -1.
Polling Interval	The interval, in seconds, between each call to drainInput() on the exercise connection. Default: 1.
Location Filtering Interval	The time, in seconds, between checking entity locations for filtering
Statistics Reporting Interval	The frequency that the broker should report statistics.
Print Statistics to Log File	If true, print the broker statistics to the log file specified in Log Filename.

4.2.3. Configuring Publication Conditions

By default, VR-Exchange does not translate some HLA objects until certain conditions are met. For example, an entity is not translated until its entity ID is a non-zero value. This is done because various mappings exist between DIS and HLA identifiers. However, there are situations in which IDs and other attributes are not specified. Turning off publication conditions allows you to translate these objects.

If a condition is set to True, an HLA object is not published until the condition is true. If a condition is set to False, the object is published without satisfying this condition.

You can set discovery criteria for the following objects:

- Aggregate:
  - Aggregate needs valid Entity ID.
  - Aggregate needs valid Location.
  - Aggregate needs valid Type.
- Designator:
  - Designator needs valid Host ID.
  - Designator needs valid Entity ID.
- Emitter System:
  - Emitter System needs valid Entity ID.
  - Emitter System needs valid Host ID.
- Entity State:
  - Entity needs valid Entity ID. (HLA only.)
  - Entity needs valid Location.
  - Entity needs valid Type.
- Environment Process:
  - Environment Process needs valid Process ID.
- Gridded Data:
  - Gridded Data needs valid Grid ID.
- IFF:
  - IFF needs valid Entity ID.
  - IFF needs valid Host ID.
- Radio Receiver:
  - Radio Receiver needs valid Entity ID.
  - Radio Receiver needs valid Host ID.
- Radio Transmitter:
  - Radio Transmitter needs valid Entity ID.
  - Radio Transmitter needs valid Host ID.
- Underwater Acoustics:
  - Underwater Acoustics needs valid Acoustic ID.
  - Underwater Acoustics needs valid Entity ID.

– Underwater Acoustics needs valid Host ID.

To configure publication conditions:

1. In the Connections window (Figure 3-1), select the connection that you want to configure.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens.
3. Select the Configure page (Figure 4-8).

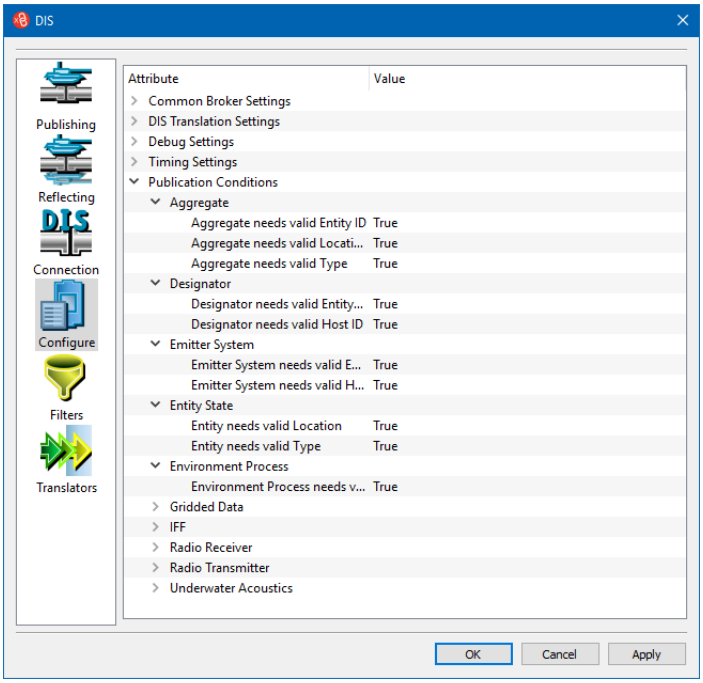

Figure 4-8. Publication conditions

4. Expand the Publication Conditions section.
5. Expand the hierarchy for the message type whose publication conditions you want to change.
6. Click the value to make it editable.
7. Select the value that you want.
8. Click OK.

4.2.4. Protocol Specific Attributes

The Configure page for the different brokers has some attribute groupings used in one or more brokers, but the attributes themselves vary by broker. The following sections describe them.

HLA Translation Settings and Timing Settings

Table 4-2 describes the HLA-specific attributes grouped under translation settings and timing settings.

Table 4-2: HLA connection attributes (Sheet 1 of 2)

Attribute	Description
HLA Translation Settings	
Fill Outgoing Entity IDs	Enables or disables filling out the entity ID (site:host:id) for all outgoing HLA entities. Use this parameter to spoof gateways that require DIS entity IDs for RPR FOM objects. Default: False.
Use Persistent Entity IDs	Specifies that entity IDs persist after disconnects. If this feature is enabled, when you replay an exercise, entities with common names will use the same entity IDs that they did in previous runs. To enable this feature, set Use Persistent Entity IDs to True. Default: False. You must also enable Fill Outgoing Entity IDs. This attribute saves entity IDs to the file *hlaPersistentEntityNumMap.csv*. Default: False.
Outgoing Site ID	The site ID to use when Fill Outgoing Entity IDs is True. Default: 3.
Outgoing Host ID	The host ID to use when Fill Outgoing Entity IDs is True. Default: 4.
Use HLA Object IDs For Marking Text	When enabled, the broker uses the HLA object ID from reflected HLA objects to set the marking text when it sends information to the Portal. This is useful for mapping non-RPR FOM-based FOMs that do not have marking text to RPR FOM-based FOMs that might rely on it. Default: False.
Use Marking Text for HLA Object IDs	When enabled, the broker uses the marking text for objects it discovers in the Portal to name the HLA objects it is sending to its federation. If this parameter is enabled, HLA objects are not published until a valid marking text is supplied. Default: False.
Use Absolute Time-stamps	Enables or disables use of absolute time stamping. Default: False.
Disable Articulated Parts Translation	Specifies whether or not to translate articulated parts. This can be used when your exercise has circular dependencies or otherwise non-standard configurations of articulated parts that might confuse the brokers.
Use Generic Translators for Whole FOM	Enables or disables use of generic translation for the entire FOM. Default: False.

Table 4-2: HLA connection attributes (Sheet 2 of 2)

Attribute	Description
Generic Objects to Translate Set	A list of HLA object classes to translate using the generic object translator. The generic object translator does not work with a FOM Mapper and the classes represented in this list must be exactly the same in all receiving FOMs. In cases where there are more than two brokers, only the two for which they are exactly the same should have this parameter set. (For information about generic translation, please see "Generic Translations," on page A-4.) Default: an empty list.
Generic Interactions to Translate Set	A list of HLA interaction classes to translate using the generic interaction translator. (For information about generic translation, please see "Generic Translations," on page A-4.) Default: an empty list.
Timing Settings	
Heartbeat	The frequency with which to send an update if there is no change. Default: -1 (means do not heartbeat).
HLA Maximum Tick Time	Specifies the maximum amount of time, in seconds, to tick the RTI before processing incoming messages. -1 = Tick the RTI until all messages are read. >0 = Tick the RTI at least HLA Minimum Tick Time, but no more than this value. Default: -1.
HLA Minimum Tick Time	Specifies the minimum amount of time, in seconds, to tick the RTI before processing incoming messages. Default: 0.0.

DIS Translation Settings, Debug Settings, and Timing Settings

Table 4-3 lists DIS-specific attributes grouped under translation settings, debug settings, and timing settings.

Table 4-3: DIS connection attributes (Sheet 1 of 2)

Parameter	Description
DIS Translation Settings	
Parse Unknown IDs	Tells the DIS Broker to try to parse a DIS ID out of a Portal Object ID if there is not already a known entity available. For example, if the Portal ID (a string) is "1:2:3", and if the DIS Broker does not already know about string "1:2:3", it tries to create a DIS ID from the string, in this case 1:2:3. Default: True.
Remap DIS IDs	Enables or disables mapping of all DIS entity IDs to match the site and host IDs specified in the configuration file. This corrects errors caused by rogue DIS objects in a DIS-to-HLA-to-DIS exercise. DIS IDs are remapped on send. Default: False.
Use Persistent Entity IDs	Specifies that entity IDs persist after disconnects. If this feature is enabled, when you replay an exercise, entities with common names use the same entity IDs that they did in previous runs. To enable this feature, set Use Persistent Entity IDs to True. You must also enable Fill Outgoing Entity IDs. This attribute saves entity IDs to the file *disPersistentEntityNumMap.csv*. Default: False.
Application Number	The DIS application number. It is used for creating entity IDs for PDUs originating from the DIS broker. It must be unique. Default: 27.
Site ID	Specifies the DIS site ID. Default: 1.
Use Absolute Timestamps	Specifies the use of absolute timestamps on PDUs or interactions. False = Do not use absolute timestamps. True = Use absolute timestamps. Default: False.
DIS Protocol Version to Send	The DIS PDU version to send, where: 4 = DIS 2.0.4 5 = IEEE 1278.1 6 = IEEE 1278.1a 7 = IEEE 1278.1-2012 Default: 5.
DIS Minimum Protocol Version to Receive	The minimum DIS PDU version to receive. Default: 7.
DIS Maximum Protocol Version to Receive	The maximum DIS PDU version to receive. Default: 6.

Table 4-3: DIS connection attributes (Sheet 2 of 2)

Parameter	Description
Disable Articulated Parts Translation	Specifies whether or not to translate articulated parts. This can be used when your exercise has circular dependencies or otherwise non-standard configurations of articulated parts that might confuse the brokers.
Debug Settings	
Debug Global ID Mappings	Enables or disables mapping of Portal object IDs to DIS entity IDs in the debug output. notifyLevel must be set to at least 2 for this to work. Default: False.
Debug Entity ID Mappings	Enables or disables mapping of Portal entity IDs to DIS entity IDs in the debug output. Default: False.
Debug Event ID Mappings	Enables or disables mapping of Portal event IDs to DIS event IDs in the debug output. Default: False.
Dump GUI to Text	Enables or disables logging of GUI updates to the log file. When enabled, the Portal outputs a line each time an update comes from the translator. This information is already displayed in the GUI, but using Dump GUI to Text makes a better record for debugging. Notify Level must be set to at least 2 for this to work. Default: False.
Timing Settings	
Heartbeat Interval	Specifies the number of seconds between heartbeats of published entities. Default: 5.0 seconds.
Timeout Interval	Specifies the number of seconds before a reflected entity times out and the DIS broker removes it. Default: 10.0 seconds.
Drain Timeout	Specifies the timeout value, in seconds, for reading PDUs from the network, as follows: -1 = No timeout. Poll network until all PDUs are read. >0 = Poll the network, reading the number of PDUs specified by Number of PDUs Read Per Tick, until all PDUs are read or the timeout value is reached, whichever comes first. Default: -1.
Number of PDUs Read Per Tick	Specifies the number of PDUs to read before checking to see if the amount of time specified by Drain Timeout has been reached. Default: 100.
Multicast Time-to-Live	Specifies a multicast time-to-live value. Default: 0 (use the system default).

DDS Translation Settings, Debug Settings, and Timing Settings

Table 4-4 lists DDS-specific attributes grouped under translation settings, debug settings, and timing settings.

Table 4-4: DDS connection attributes (Sheet 1 of 2)

Parameter	Description
DDS Translation Settings	
Parse Unknown IDs	Tells the DIS Broker to try to parse a DDS ID out of a Portal Object ID if there is not already a known entity available. For example, if the Portal ID (a string) is "1:2:3", and if the DDS Broker does not already know about string "1:2:3", it tries to create a DDS ID from the string, in this case 1:2:3. Default: True.
Fill Outgoing Entity IDs	Enables or disables filling out the entity ID (site:host:id) for all outgoing HLA entities. Use this parameter to spoof gateways that require DIS entity IDs for RPR FOM objects. Default: False.
Use Persistent Entity IDs	Specifies that entity IDs persist after disconnects. If this feature is enabled, when you replay an exercise, entities with common names use the same entity IDs that they did in previous runs. To enable this feature, set Use Persistent Entity IDs to True. You must also enable Fill Outgoing Entity IDs. This attribute saves entity IDs to the file *disPersistentEntityNumMap.csv*. Default: False.
Use DDS Object ID for Marking Text	Fill the Marking Text field with the DDS Object ID.
Use Marking Text for DDS Object ID	Fill the DDS Object ID with the marking text if it is properly formatted.
Guise is Entity Type	Use the guise attribute as the entity type.
Force Updates	Force all objects to update even if they have not changed.
Debug Settings	
Debug Global ID Mappings	Enables or disables mapping of Portal object IDs to DS entity IDs in the debug output. notifyLevel must be set to at least 2 for this to work. Default: False.
Debug Entity ID Mappings	Enables or disables mapping of Portal entity IDs to DDS entity IDs in the debug output. Default: False.
Dump GUI to Text	Enables or disables logging of GUI updates to the log file. When enabled, the Portal outputs a line each time an update comes from the translator. This information is already displayed in the GUI, but using Dump GUI to Text makes a better record for debugging. Notify Level must be set to at least 2 for this to work. Default: False.

Table 4-4: DDS connection attributes (Sheet 2 of 2)

Parameter	Description
Timing Settings	
DDS Tick Time	The tick rate for the DDS broker.
Object Throttle Rates	The DDS global object throttle rate.

4.3. Configuring Network Connections

Each broker connection dialog box has a Connection page that lets you configure the network connections for the broker. The parameters are protocol-specific.

To configure a broker's network connection:

1. In the Connections window (Figure 3-1), select the connection that you want to configure.
2. Choose **Connection** → **Edit Connection Information** or double-click the connection. The Connection Information dialog box opens.
3. Select the Connection page. Figure 4-9 illustrates the Connections page for a DIS broker.

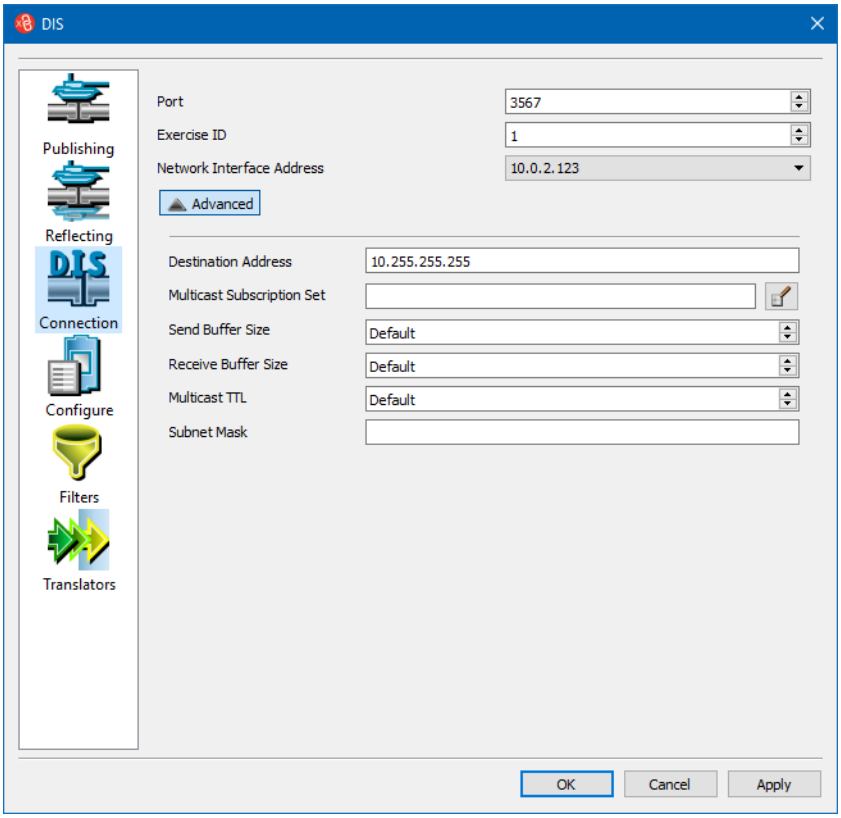

Figure 4-9. DIS Network Connections page

4. Edit the parameters. The parameters are described in the following sections.

4.3.1. HLA Network Connection Parameters

Table 4-2 describes the HLA-specific parameters that you can configure for connections that use the HLA broker.

Table 4-5: HLA network connection parameters (Sheet 1 of 2)

Attribute	Description
HLA Execution Name	Specifies the name of the HLA federation execution that this broker will join. Default: VR-Link.
HLA FED File Name	Specifies the name of the FED file being used for the federation execution. It must be the complete name including the three character extension (*.fed*). Default: VR-Link.fed.
Federate Type	Specifies the HLA Federate type. Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore. Default: VR-Exchange.
FOM Mapper Library Name	Specifies the name of a FOM mapper library, if you are using one. If you want to use one of the FOM Mappers supplied with VR-Exchange, leave this attribute blank and specify a FOM Mapper for the Built In FOM Mapper Attribute. Default: "".
FOM Mapper Initialization Data	Specifies a string that is passed to the FOM mapper library specified in FOM Mapper Library Name. This data is optional. Default: "".
Built In FOM Mapper	If the federation is using the RPR FOM, specifies the version, and for RPR FOM 2, draft 17, the revision. RPR FOM revision 2 for version 2.0017 fixes an encoding and decoding size issue with object ID lists to match RPR FOM 2 draft 17. The encoding and decoding issues affected the Aggregates, JammerBeams, and RadarBeams objects and the RemoveObjectRequest, AttributeChangeRequest, and ActionRequestToObject interactions. The built-in FOM Mappers are: ◆ RPR FOM 1.0 ◆ RPR FOM 2, draft 14 ◆ RPR FOM 2, draft 17, Revision 1 ◆ RPR FOM 2, draft 17, Revision 2. Default: RPR FOM 1.0.
Use HLA Advisories	Specifies whether the broker should use RTI advisory messages when determining what attributes to update. False = Send attribute updates regardless of whether or not someone has subscribed to the attribute with the RTI. True = Send attribute updates only if someone has subscribed to the attribute. Default: False.

Table 4-5: HLA network connection parameters (Sheet 2 of 2)

Attribute	Description
Additional Connection Settings for HLA Evolved	
Federate Name	The Federate Name is a unique name that can be assigned to the federate. No two federates within the same federation can have the same name. If no name is specified, the RTI automatically assigns a unique name for the federate.
HLA Evolved FOM Module Names	HLA Evolved allows federates with different FOMs to interoperate without merging their FOMs. This is done by loading FOM modules.
Local Settings Designator	A string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage. This setting is optional for the MAK RTI. If you specify it, it overrides the RID file settings.

4.3.2. DIS Network Connection Parameters

Table 4-3 lists parameters that you can configure for network connections that use the DIS broker.

Table 4-6: DIS network connection parameters (Sheet 1 of 2)

Parameter	Description
Port	Specifies the port to receive network PDUs from and send them to. Default: 3000.
Exercise ID	Specifies the DIS exercise ID. Default: 1.
Network Interface Address	Specifies the IP address of the network device to use for sending DIS PDUs.
Destination Address	Specifies the address of outgoing DIS PDUs. If set to 0.0.0.0, the default broadcast address is used.
Multicast Subscription Set	Specifies a list of multicast addresses the DIS broker should subscribe to for DIS communication. Default: an empty list.
Send Buffer Size	Sets the socket's send buffer size. Default: 0.
Receive Buffer Size	Sets the DIS socket's buffer received size. If you think the broker is dropping incoming packets during periods of heavy network traffic, increasing this value might help. A typical default value is 50000. If you set the size to zero, the DIS broker uses the default size for the operating system. Default: 0.

Table 4-6: DIS network connection parameters (Sheet 2 of 2)

Parameter	Description
Multicast TTL	Specifies the number of routers that a message can pass through when PDUs are sent to multicast addresses.
Subnet Mask	VR-Exchange can usually figure out the subnet mask from the IP address. However, if it does not do this correctly, you can explicitly specify the subnet mask.

4.3.3. DDS Network Connection Parameters

The DDS broker has two network connection parameters – Partition Name and Domain ID.

Domain ID is an integer that represents an address space for DDS applications. DDS applications must join the same DDS Domain to communicate.

Partition Name specifies the DDS partition to join using the RTS. Partitions can be used to control which DataWriters can communicate with DataReaders within a domain.

4.4. Specifying Filters in the Connection Information Dialog Box

You can filter out objects by their object type, object name, marking text, or geospatial location. Filtering by entity type and marking text only applies to entities and aggregates. The filtering mechanism supports use of wildcards.

When you filter by geospatial location, you supply a latitude, a longitude, and a radius. All entities that are outside of the radius get filtered out.

This section explains how to add and remove filters in the Connection Information dialog box. You can also do so dynamically in the Objects window. For details, please see "Filtering Objects in the Objects Window," on page 3-14.

To filter an object:

1. Select the broker connection on which you want to configure a filter.
2. Choose **Connection** → **Edit Connection Information** or double-click the connection. The Connection Information dialog box opens.
3. Select the Filter page (Figure 4-10). It displays any filters that you have set in the Objects window or in this dialog box.

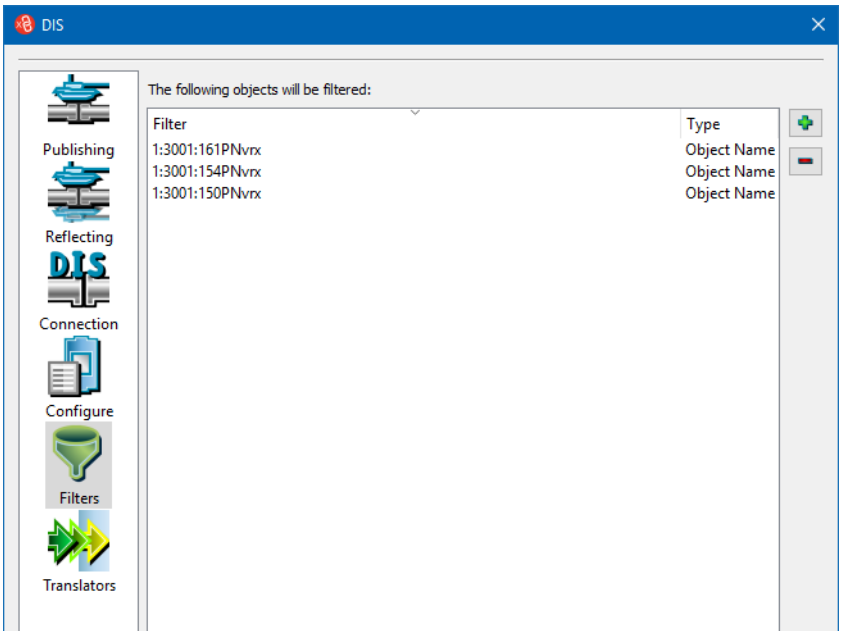

Figure 4-10. Filter page

4. Click the Add icon (+). The New Filter dialog box opens (Figure 4-11).

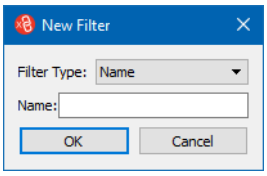

Figure 4-11. New Filter dialog box

5. Select the filter type in the Filter Type drop-down list. The dialog box redisplays and requests the type of information needed for the selected filter.
6. Enter the filter value. If you chose Name or Marking Text as the filter type, type the appropriate text for the object you want to filter. You can use wildcards such as *, ?, and [abcd] to define a generalized ID.

   If you chose Type, VR-Exchange automatically inserts an enumeration value (all zeros) in the field. Type the enumeration for the object you want to filter. (If you mouse over the entity icon in the Type column of the Objects window, VR-Exchange displays the entity type enumeration for the object. This information may be helpful in deciding the enumeration you want to use when filtering by entity type.)

   If you chose Geospatial Location (Figure 4-12), provide the latitude and longitude of the location, in degrees, minutes, seconds, and the radius around it outside of which entiites should be filtered out.

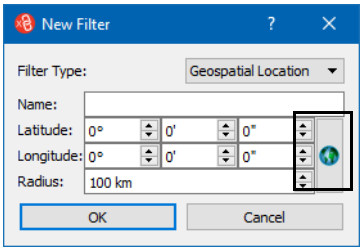

Figure 4-12. Geospatial Location filter

You can also specify the location graphically, as follows:

a. Click the map button (Figure 4-12). A map of the world opens.
b. Draw a circle to define the area you want to filter (Figure 4-13).

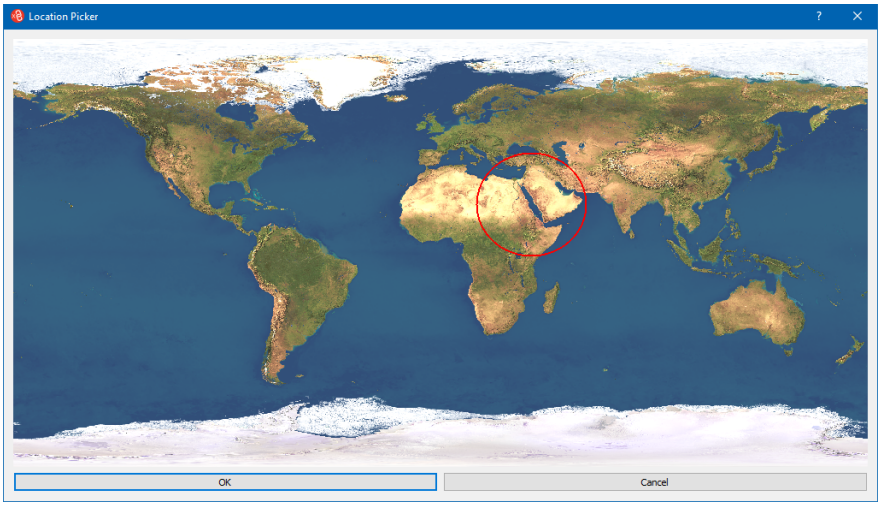

Figure 4-13. Location Picker window

c. Click OK.

7. Click OK. The new filter is added to the list.

4.4.1. Editing a Filter

To edit a filter:

1. Select the broker connection for which you want to edit a filter.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens.
3. Select the Filter page (Figure 4-10).
4. Double-click the filter that you want to edit. The New Filter dialog box opens (Figure 4-11).
5. Edit the values as desired.
6. Click OK.

4.4.2. Removing a Filter

When you delete a filter in the Connection Information dialog box, its status in the Objects window is updated. However, there may be some latency between the two events.

To delete a filter:

1. Select the broker connection from which you want to remove a filter.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens.
3. Select the Filter page (Figure 4-10).
4. Select the filter that you want to delete.
5. Click the Delete icon (−).

4.5. Specifying Which Translators to Use

Each broker has a set of translators. For each translator, you can specify whether or not it reads from the network and writes to the network.

> When you enable or disable a translator, you must restart the broker for the change to take effect.

To specify which translators a connection should use:

1. In the Connections window (Figure 3-1), select the connection that you want to configure.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens.
3. Select the Translators page (Figure 4-14).

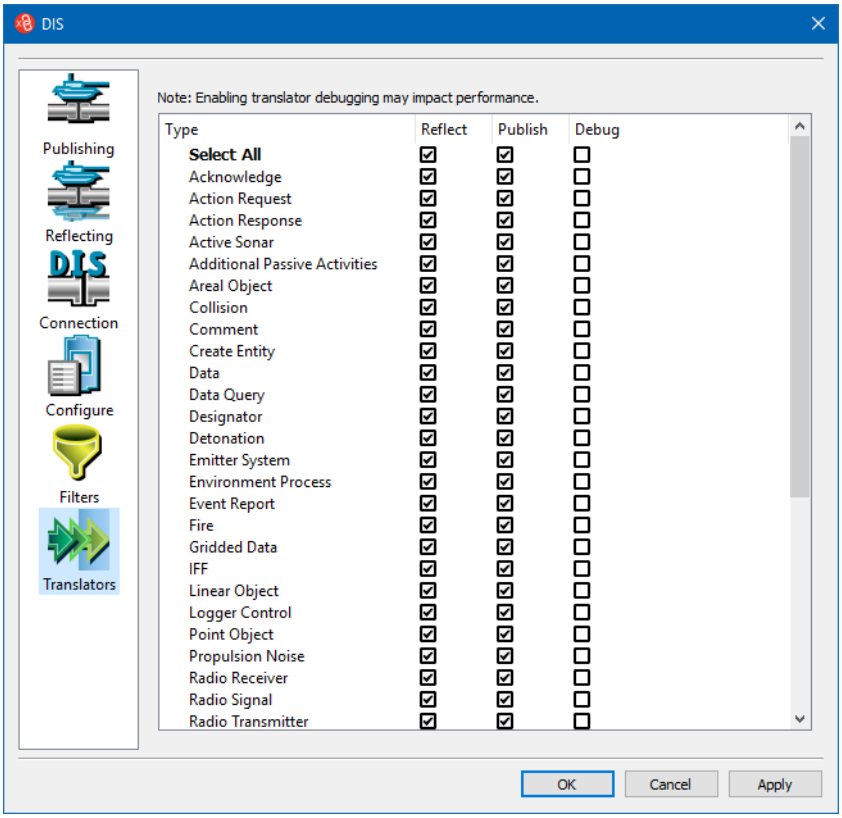

Figure 4-14. Translators page

4. In the Reflect column, select the check box for each translator that you want the broker to read from the network. Clear the check box for each translator that you do not want to read.

5. In the Publish column, select the check box for each translator that you want the broker to write to the network. Clear the check box for each translator that you do not want to write.
6. In the Debug column, select the check box for a translator if you want VR-Exchange to send debugging messages for this translator. If debugging is enabled, the notify level must be at least 2.
7. To select or clear all check boxes in a column, select or clear the Select All check box at the top of the list.
8. Click OK.

4.6. Viewing Publishing and Reflecting Lists

You can view lists of the objects that a connection is publishing and reflecting. You can open information windows for multiple connections at the same time.

- The Publishing page displays the objects that the connection has discovered from the Portal and published to its simulation network.
- The Reflecting page displays the objects that the connection has discovered on its simulation network and published to the Portal.

To view connection information:

1. In the Connections window (Figure 3-1), select the connection whose details you want to display.
2. Choose **Connection** → **Edit Connection Information**, or double-click the connection. The Connection Information dialog box opens.
3. Select the Publishing page or the Reflecting page (Figure 4-15).

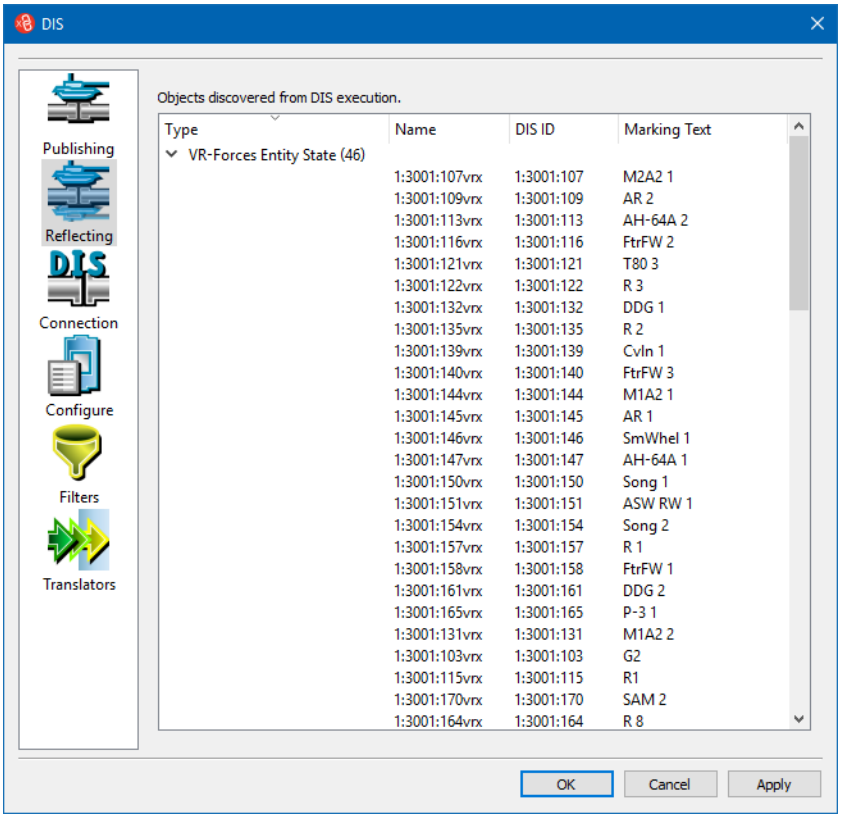

Figure 4-15. HLA connection Reflecting page

5. VR-Exchange Tutorials

This chapter has tutorials for common VR-Exchange use cases.

5.1. Examples of Using VR-Exchange

This section has two examples of using VR-Exchange. The first shows how to have a DIS application interact with an HLA application on the same computer. The second shows how to connect two federates using different RTI versions. In both tutorials we use the MAK Data Logger to send data through the connection and VR-Vantage Stealth to receive the data. You can, of course, use any application that you want to use when you try these tutorials.

5.2. An HLA to DIS Example

In this example, we use VR-Exchange to connect the output of a MAK Data Logger running in HLA to VR-Vantage Stealth running in DIS. (The procedure assumes you know how to use these applications.)

To connect a DIS application to an HLA application:

1. Start the HLA 1516 Evolved RPR FOM 2.0 version of the MAK Logger and load an HLA Logger file, such as *makland2018-HLAe-RPR20.lgr*.
2. Start VR-Vantage Stealth. Load the *Makland.mtf* terrain and connect to DIS.
3. Start VR-Exchange.
4. Open the DIS connection and make sure that it is using the same port value as VR-Vantage. If necessary, change the port value and restart the connection:
   a. In the Connections window, double-click the DIS connection. The Connection Information dialog box opens.
   b. Select the Connection page.
   c. If necessary, change the port number.
   d. Click OK.
5. The connections should be connected. If they are not, enable them.
6. Play the Logger file. You should see objects and interactions listed in VR-Exchange.
7. Observe VR-Vantage Stealth to see that it is receiving the output from VR-Exchange (Figure 5-1).

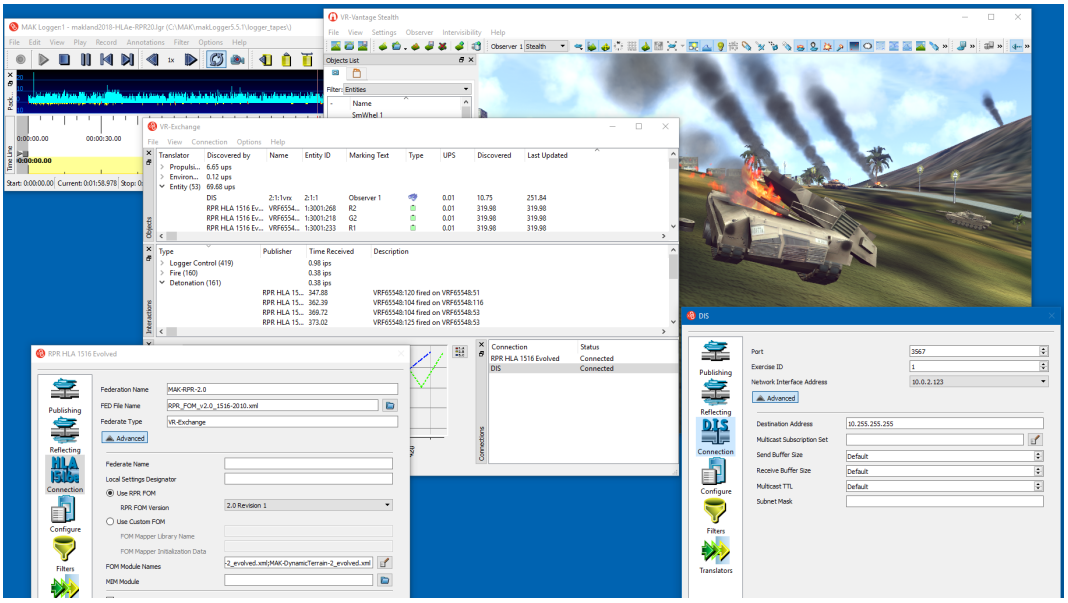

Figure 5-1. HLA to DIS example

5.3. Connecting Two Different RTIs

An important use for VR-Exchange is to connect applications running different RTIs or different versions of the same RTI. This example shows how to connect applications that are using different versions of the MAK RTI. The applications could be running on the same computer (Figure 5-2) or on separate computers (Figure 5-3). In either case, you must install the two RTIs on the computer that is running VR-Exchange and must configure VR-Exchange brokers the same way.

The RTIs must be built with the same compiler as VR-Exchange.

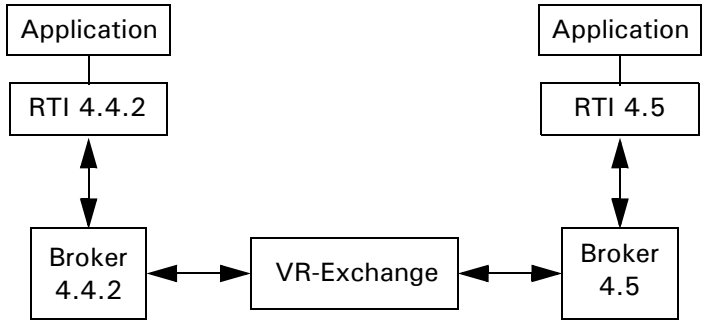

Figure 5-2. VR-Exchange configuration - one computer

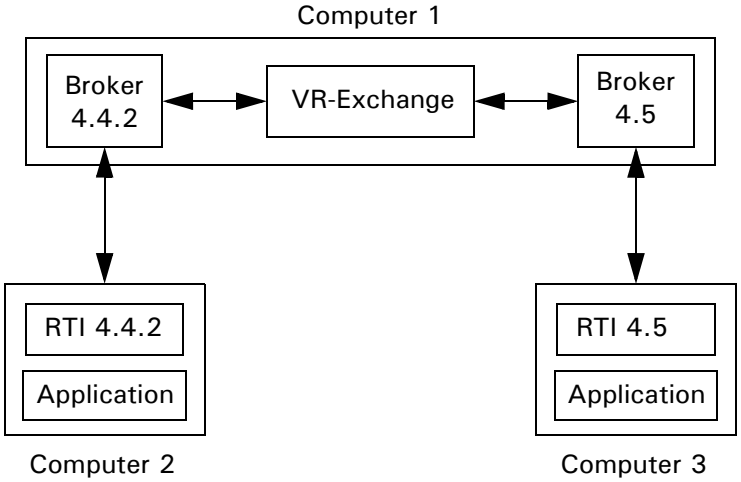

Figure 5-3. VR-Exchange configuration - three computers

The overall procedure is as follows:

1. Install the RTIs.
2. Create brokers for the two RTIs.
3. Enable the brokers.
4. Run the applications.

5.3.1. Installing the RTIs

For this tutorial, we use MAK RTI 4.4.2 vc12 and MAK RTI 4.5 vc12 with a vc12 version of VR-Exchange. The key to this procedure is to set the environment variables for each RTI in the VR-Exchange broker, not as part of the installation.

To install the RTIs:

1. Install the RTIs using their installers. When you are prompted to specify the RTI as the default, clear the check box. Do not set either RTI as the default.
2. Check your environment variables and make sure that the following variables are not set:
   - RTI_RID_FILE.
   - RTI_ASSISTANT_PORT.
   - PATH. Be sure that there are no RTIs listed in your PATH.

The default HLA 1516 broker included with VR-Exchange will generate error messages when you start up because it will not be able to find an RTI in the search paths. You can ignore the error messages.

5.3.2. Add the Brokers

You must add a broker for each RTI.

To add a broker:

1. Choose **Connection** → **Add New Connection**. The Add New Connection dialog box opens (Figure 5-4).
2. In the Connection Name box, type a name for the connection, for example RTI 4.4.2.
3. In the Broker Executable box, select a broker from the drop-down list or type the name of a broker executable. For this tutorial, select rprHla13Broker.

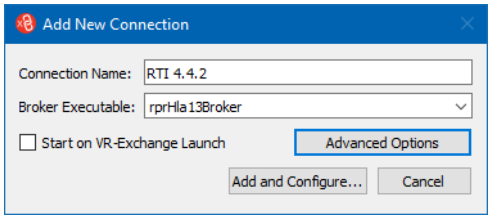

Figure 5-4. Add New Connection dialog box

4. Click Advanced Options. The dialog box expands (Figure 5-5).

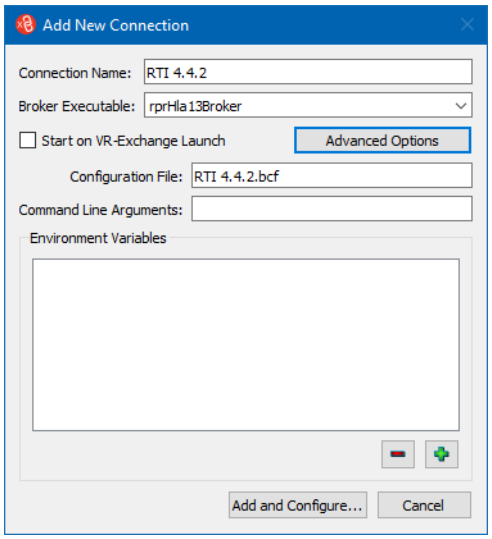

Figure 5-5. Add New Connection dialog box, Advanced Options

5. Specify the environment variables for the broker:
   - RTI_RID_FILE. The RID file for MAK RTI 4.4.2.
   - PATH. The full system path with the RTI *./bin* directory prepended. You can use variable substitution (on Windows, %PATH%; on Linux, $PATH or ${PATH}). For example: `C:\MAK\makRti4.4.2\bin;%PATH%`
   - RTI_ASSISTANT_PORT. The port the RTI assistant should use.

   a. Click the Add button (+) under the Environment Variables window. The Environment Variable dialog box opens (Figure 5-6).

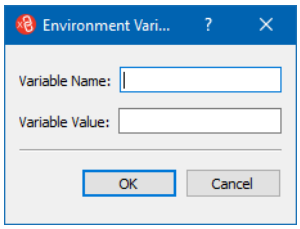

Figure 5-6. Environment Variable dialog box

b. In the Variable Name box, type the name of the environment variable you want to add, for example, `RTI_RID_FILE`.

c. In the Variable Value box, type the value `C:\MAK\makRti4.4.2\rid.mtl`.

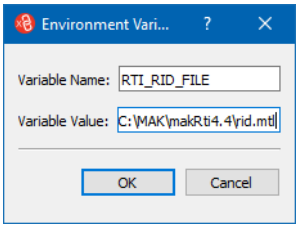

Figure 5-7. Completed Environment Variable dialog box

d. Click OK.

e. Repeat the procedure for each environment variable. Use 6003 for the port.

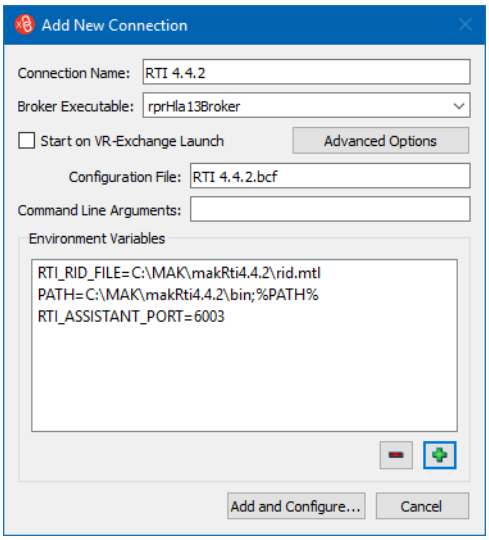

Figure 5-8. Add New Connection dialog box, completed

6. Click Add and Configure.

7. Repeat the previous steps to add a broker for MAK RTI 4.5. In the Broker Executable list, select rprHla1516eBroker. Use a different port for the RTI_ASSISTANT_PORT environment variable, such as 6004.
8. The new brokers are now listed in the Brokers window (Figure 5-9).

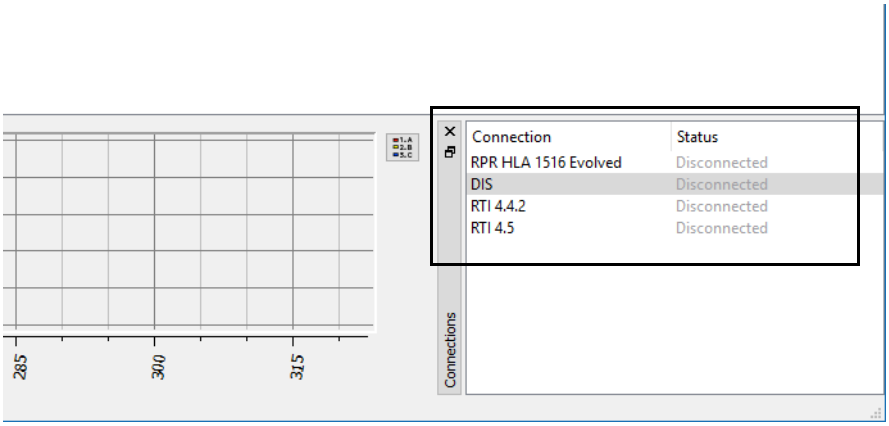

Figure 5-9. New brokers

5.3.3. Connect the Brokers

Each broker needs to connect to an RTI connection using the correct version of the RTI. They must have different ports. We will create these connections when we connect the broker.

The configuration process described in this section is for the MAK RTI. If you are using a different RTI, the process will probably be different.

To connect the brokers:

1. In the Connections window, right-click the RTI 4.4.2 broker. A menu is displayed.
2. Choose **Enable Connection**. The Choose RTI Connection dialog box opens. It lists the available RTI connections.

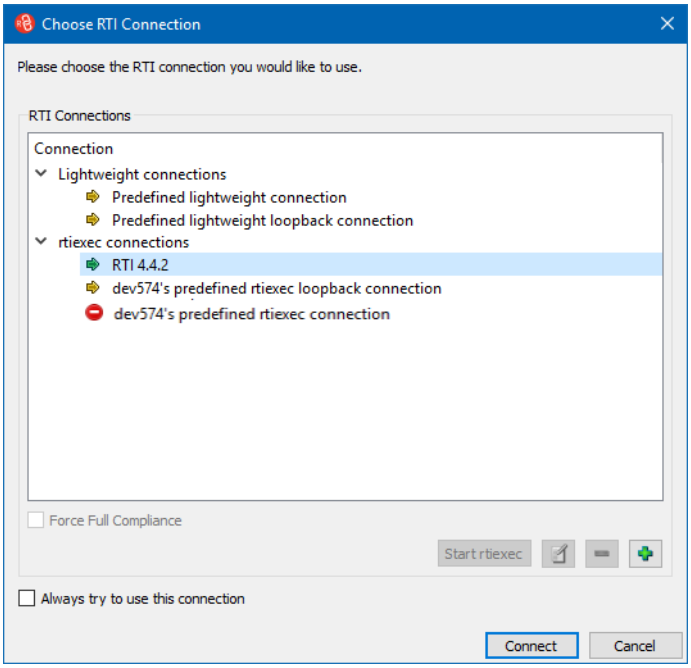

Figure 5-10. Choose RTI Connection

3. Select the predefined rtiexec connection for your computer.
4. Click the Add button (+). The Add New RTI Connection dialog box opens (Figure 5-11). The Connection Name field has the name of your computer with the number 1 added.

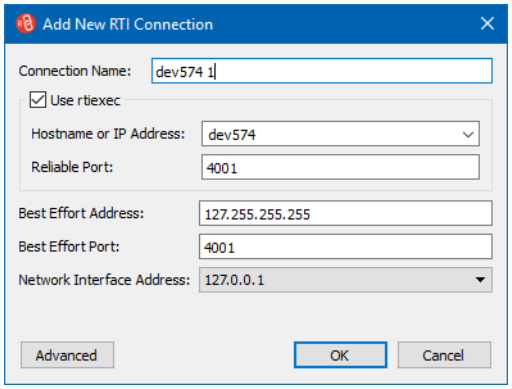

Figure 5-11. Add New RTI Connection

5. Optionally, change the name of the connection, for example, RTI 4.4.2.
6. Change the Reliable Port value. Add a 4 to the default value, since this connection is for RTI 4.4.2 (making it 40014).
7. Change the Best Effort Port value by adding a 4 to the default value (making it 40014).

This procedure suggests specific port numbers. There is nothing special about these numbers. The important thing is that the port numbers be different for the two RTIs.

8. If the Advanced Forwarder Options group box is not displayed, click Advanced.
9. Change the Forwarder Port value to 5004. The completed connection should look similar to Figure 5-12.

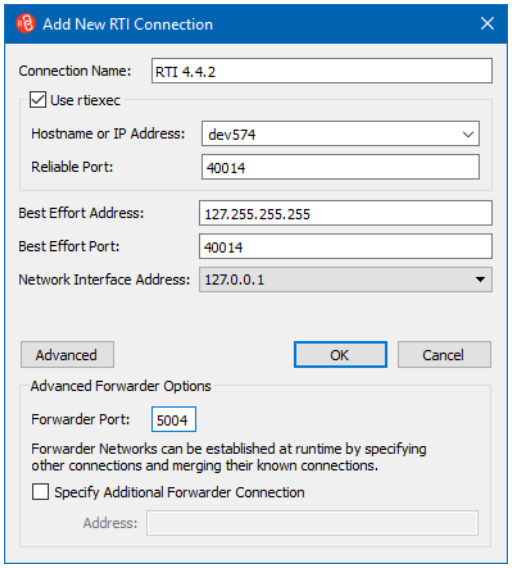

Figure 5-12. New RTI connection

10. Click OK.
11. In the Choose RTI Connection dialog box, select the new connection (Figure 5-13).

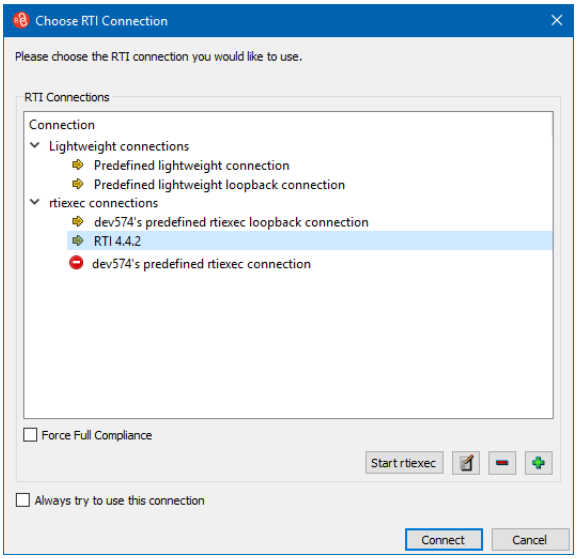

Figure 5-13. Select new RTI connection

12. Click Connect. The broker connects to the RTI.
13. Repeat this procedure for the RTI 4.5 broker. Add a 5 to the port values and change the forwarder port to 5005.

The Connections panel now shows the two brokers are connected .

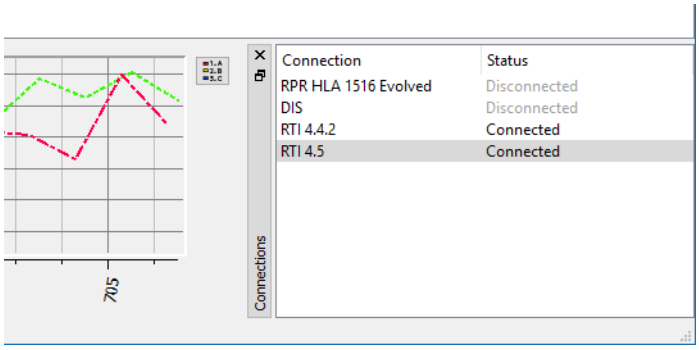

Figure 5-14. Connected brokers

5.3.4. Run the Appplications

If you are running your applications on separate computers, as shown in Figure 5-3, start the applications, for example a MAK Data Logger on one computer playing an HLA1516e recording, and VR-Vantage on the second computer to display the simulation using an HLA 1516e connection. VR-Exchange should pick up the network traffic from the Logger using its RTI 4.4.2 broker and send it out using its RTI 4.5 connection to VR-Vantage.

If you are running the applications on the same computer as VR-Exchange, the setup is a bit more complicated. You must set up different RTI environments and run the applications from them.

To run applications on one computer using different versions of the MAK RTI:

1. Open up two console windows. One is for MAK RTI 4.4.2 and the other for MAK RTI 4.5.
2. In one console window, set the environment to match the RTI 4.4.2 broker, as follows:

```
>SET RTI_RID_FILE=C:\MAK\makrti4.4.2\rid.mtl
>SET RTI_ASSISTANT_PORT=6003
>SET PATH=C:\MAK\makRTi4.4.2\bin;%PATH%
```

3. In the second console window, set the environment to match the RTI 4.5 broker, as follows:

```
>SET RTI_RID_FILE=C:\MAK\makrti4.5\rid.mtl
>SET RTI_ASSISTANT_PORT=6004
>SET PATH=C:\MAK\makRTi4.5\bin;%PATH%
```

If you do this regularly, you may want to write scripts to set the environment.

4. In the RTI 4.4.2 window, run the MAK Data Logger (or the federate of your choice):

   ```
   >cd C:\MAK\makLoggerx.x\bin64
   >loggerhla13
   ```

   The Choose RTI Connection dialog box opens. Select the RTI 4.4.2 connection and click Connect.

5. In the RTI 4.5 window, run VR-Vantage (or the federate of your choice):

   ```
   >cd C:\MAK\vrvantagex.x\bin64
   >vrvstealth
   ```

   Connect to HLA 1516e. The Choose RTI Connection dialog box opens. Select the RTI 4.5 connection and click Connect.

6. Load an HLA 13 recording in the Logger and run it. In VR-Exchange you will see that it is receiving data from the Logger (Figure 5-15). In VR-Vantage, load the appropriate terrain and connect to HLA1516e. You will see the traffic that VR-Exchange is sending to the network using RTI 4.5.

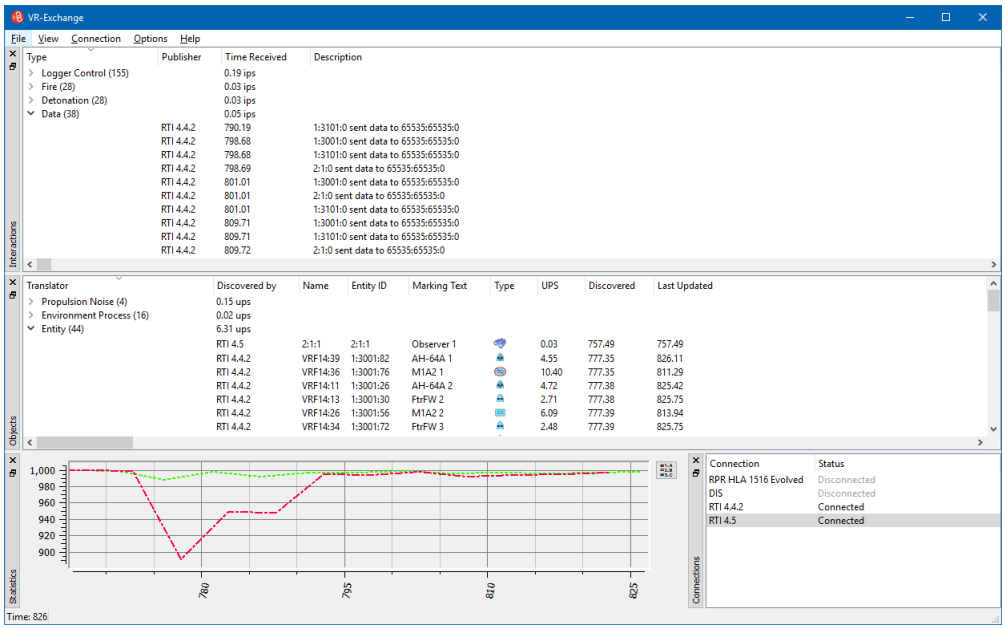

Figure 5-15. VR-Exchange routing RTI 4.4.2 traffic to RTI 4.5

A. Broker Details

This appendix lists the translators provided by each broker and lists their command-line options. You can add command-line options to a connection in the Add Broker dialog box and the Edit Broker dialog box.

A.1. The HLA Brokers

VR-Exchange includes HLA brokers for the HLA 1.3 specification, for the SISO Standard DLC C++ API for IEEE 1516 specification, and for the HLA Evolved specification. They have built-in support for the RPR FOM using the VR-Link FOM Mappers and FED files provided. They can support other FOMs through the use of user-written FOM Mappers or generic translation.

The brokers work identically and support the same configurations, so for simplicity, in the remainder of this manual we refer to them generically as the HLA broker.

A.1.1. HLA Translators

The HLA broker is composed of translators (described in "Translators," on page 1-3). Each translator is responsible for a specific FOM object class, or interaction class. Table A-1 lists the translators in the HLA broker. The HLA broker can use a VR-Link FOM Mapper to modify the translations of any of the translators. For information about FOM mapping, please see *VR-Link Developers Guide*.

Table A-1: HLA broker translators

Translator	RPR FOM Class
Acknowledge	InteractionRoot.Acknowledge
Action Request	InteractionRoot.ActionRequest
Action Response	InteractionRoot.ActionResponse
Active Sonar	ObjectRoot.EmbeddedSystem.Underwater-AcousticsEmission.ActiveSonar
Additional Passive Activities	ObjectRoot.EmbeddedSystem.Underwater-AcousticsEmission.AdditionalPassiveActivities
Aggregate	ObjectRoot.BaseEntity.AggregateEntity
Areal Object	ObjectRoot.EnvironmentObject.ArealObject
Collision	InteractionRoot.Collision
Comment	InteractionRoot.Comment
Create Entity	InteractionRoot.CreateEntity
Data	InteractionRoot.Data
Data Query	InteractionRoot.DataQuery
Designator	ObjectRoot.Designator
Detonation	InteractionRoot.MunitionDetonation
Emitter System	ObjectRoot.EmitterBeam ObjectRoot.EmbeddedSystem.EmitterSystem

Table A-1: HLA broker translators

Translator	RPR FOM Class
Entity State	ObjectRoot.BaseEntity.PhysicalEntity.Platform
Environment Process	ObjectRoot.EnvironmentProcess
Event Report	ObjectRoot.EventReport
Fire	InteractionRoot.WeaponFire
Gridded Data	ObjectRoot.GriddedData
IFF	ObjectRoot.EmbeddedSystem.IFF
Linear Object	ObjectRoot.EnvironmentObject.LinearObject
Point Object Translator	ObjectRoot.EnvironmentObject.PointObject
Propulsion Noise	ObjectRoot.EmbeddedSystem.Underwater-AcousticsEmission.PropulsionNoise
Radio Receiver	ObjectRoot.EmbeddedSystem.RadioReceiver
Radio Signal	InteractionRoot.RadioSignal
Radio Transmitter	ObjectRoot.EmbeddedSystem.RadioTransmitter
Remove Entity	InteractionRoot.RemoveEntity
Set Data	InteractionRoot.SetData
Start	InteractionRoot.StartResume
Stop	InteractionRoot.StopFreeze

A.1.2. Generic Translations

The HLA broker has a special translator called a generic translator. Each broker can configure multiple generic translators, each responsible for one specific object or interaction class. You can also specify that all translations be done generically, using none of the translators supplied with VR-Exchange. Generic translation allows you to filter parts of a FOM. It also allows you to create RTI-to-RTI bridges if the federations being bridged use the same FOM.

The generic translator does not actually do any translation. It copies each attribute or parameter byte-for-byte from and to the Portal, and publishes it to the network.

Using the same configuration as Figure 1-2, Figure A-1 illustrates how a generic translator works. This time, suppose that these FOMs do not have a translator for vehicle names. The leftmost federation sends a "lorry" message. The broker simply sends the string "lorry" to the VR-Exchange shared memory area. The second federation reads the string "lorry" from the Portal. This string is meaningful to the FOM, which is identical to that of the first federation, so it sends the "lorry" message. The third federation does not have a generic translator, so it ignores the string.

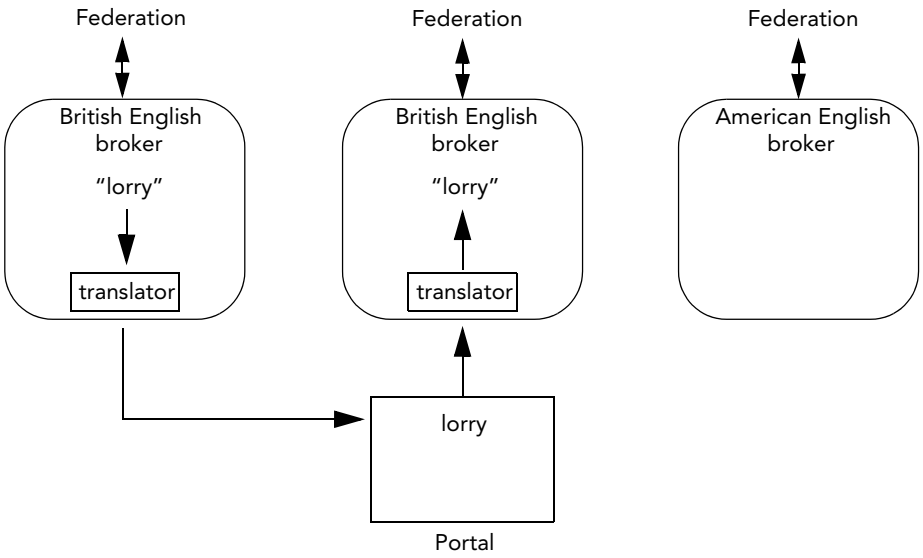

Figure A-1. How generic translators work

! To use generic translation, the sending and receiving federations must configure the same generic translators.

Generic translation only works with the HLA brokers, and only with one API at a time. This means:

- Generic messages do not work between the different HLA specifications.
- The object or interaction class must be exactly the same in all connected FOMs.
- You cannot use a FOM Mapper with generic translators.

Translating Individual Classes Generically

Translating individual classes generically is useful when you only want to translate parts of a FOM or you want to supplement the built-in translators. For example, if your exercise includes an interaction for which VR-Exchange does not have a translator, you can add a generic translator for that interaction. Bear in mind the restrictions on generic translations listed in the previous section.

To translate individual classes generically, you add the class names of the objects you want to translate to the appropriate collection in a connection's configuration. For example, you might add ObjectRoot.EnvironmentProcess to the Generic Objects to Translate Set attribute.

Translating a FOM Generically

Translating all FOM classes generically is useful if you have two or more federations that use the exact same FOM, but perhaps different RTI's. When you enable generic translation for all FOM classes, the HLA broker creates a generic translator for all objects and interactions in the FOM (except MOM data, and anything that starts with RTIPrivate). This mode prevents any built-in translators from starting.

To enable generic translation for the entire FOM, set the Use Generic Translators for Whole FOM attribute to True.

A.1.3. HLA Broker Command-Line Options

The HLA broker command-line options override the settings in a connection's configuration file. The HLA broker has the following command-line syntax:

```
{hla13Broker | hla1516Broker | hla1516eBroker} [
  -- | --ignore_rest
  --brokerName name
  --controlQueueName name
  --customFom
  --debuggedTranslators translator
  --disabledTranslators translator
  -E | --dumpEnvironment
  (-f | --fomMapperLibName) library
  --fomMapperInitData data
  --fomModules module (HLA Evolved only)
  (-F | --fedFileName) file
  -G | --noGui
  -h | --help
  --interactionQueueName name
  (-l | --configFile) filename
  (-L | --brokerLogFileName) logfile_name
  (-n | --notifyLevel) level
  -N federate_name (HLA Evolved only)
  --objectQueueName name
  -O | --oneToOneMapping
  (-p | --federateType) type
  --pluginDirectory directory
  -qtTranslatorLocale filename
  --rprFomRevision revision
  --rprFomVersion version
  --showInfoDialog
  (-S | --localSettingsDesignator) designator (HLA Evolved
  only)
  (-t | --tcpPort) port
  -v | --version
  (-x | --execName) execution]
```

where:

`(-- | --ignore_rest)` tells the broker to ignore the arguments following this flag.

`--brokerName` *`name`* specifies the name of the broker. The broker name must be unique among all brokers used. Default: Broker.

`--controlQueueName` *`name`* specifies the name of the shared memory control queue.

`--customFom` specifies3 that the fomMapperLibName and fomMapperInitData values should be used to initialize a custom FOM instead of using the RPR FOM.

`--debuggedTranslators` *`translator`* specifies translators for which you want to enable debugging.

`--disabledTranslators` *`translator`* specifies translators that you want to disable. Accepted multiple times.

`(-E | --dumpEnvironment)` print all of the environment variable settings to the log file.

`(-f | --fomMapperLibName)` *`library`* specifies the FOM mapper library, if used.

`--fomModules` *`module`* specifies a FOM module. This can be used multiple times. (HLA Evolved only.)

`(-F | --fedFileName)` *`file`* specifies the FED file name. Default: *VR-Link.fed*.

`--fomMapperInitData` *`data`* specifies FOM Mapper initialization data, if it is required.

`(-G | --noGui)` specifies that the VR-Exchange Portal GUI should not be run.

`(-h | --help)` displays usage information and exits.

`--interactionQueueName` *`name`* specifies the name of the shared memory interaction queue.

`(-l | --configFile)` *`filename`* specifies the name of the configuration file for the specified broker. Default: *broker.xml*.

`(-L | --brokerLogFileName)` *`filename`* specifies the name of the log file for the broker.

`(-n | --notifyLevel)` *`level`* specifies the output level for the broker's log file. Value values are 0 – 4. As the level increases, so does the output generated. Default: 2. The meanings of the various levels are described in "VR-Exchange Command-Line Options," on page 3-4.

`(-N | --federateName)` *`federate_name`* specifies the name of the HLA broker federate. (HLA Evolved only.)

`--objectQueueName` *`name`* specifies the name of the shared memory object queue.

`(-O | --oneToOneMapping)` forces the broker to send an update every time it receives an update from the portal, even if normal rules would dictate that an update is not necessary.

`(-p | --federateType)` *`type`* specifies the federate type. Federate Type can be used to distinguish different categories of federates, and its only practical use is in save-and-restore.

`--pluginDirectory` *`directory`* specifies the location of plug-ins.

`--rprFomRevision` *`revision`* specifies the revision of the RPR FOM version, for example, RPR FOM 2, revision 17.

`--rprFomVersion` *`version`* specifies the RPR FOM version. Default: 1.0.

`--showInfoDialog` specifies that the connection information dialog box be displayed at startup.

`(-S | --localSettingsDesignator)` *`designator`* specifies a string that is passed to the RTI during federate connect when using HLA Evolved. It is usually used to provide additional RTI configuration, but every RTI uses it differently. Please consult your RTI documentation for proper usage. This attribute is optional in the MAK RTI. You can use it to override the RID file settings. (HLA Evolved only.)

`(-t | --tcpPort)` *`port`* specifies the TCP port to use to maintain connectivity with the Portal. If the Portal dies, then the broker will know by testing this port. If this argument set to 0, the broker does not try to connect. Default: 5111.

`(-v | --version)` displays version information and exits.

`(-x | --execName)` *`execution`* specifies the HLA federation execution. Default: “VR-Link”.

A.2. The DIS Broker

The DIS broker provides comprehensive translation of DIS PDUs. It receives DIS 4, 5, 6, and 7. It sends DIS 5.

Table A-2 lists the translators supported by the DIS broker.

Table A-2: DIS broker translators

Translator	PDU
Acknowledge	Acknowledge PDU
Action Request	Action Request PDU
Action Response	Action Response PDU
Active Sonar	Underwater Acoustics PDU
Additional Passive Activities	Underwater Acoustics PDU
Aggregate	Aggregate PDU
Areal Object	Areal Object PDU
Collision	Collision PDU
Comment	Comment PDU
Create Entity	Create Entity PDU
Data	Data PDU
Data Query	Data Query PDU
Designator	Designator PDU
Detonation	Detonation PDU
Emitter Systems	Electromagnetic Emission PDU
Entity State	Entity State PDU
Environment Process	Environmental Process PDU
Event Report	Event Report PDU
Fire	Weapon Fire PDU
Gridded Data	Gridded Data PDU
IFF	IFF/ATC/NAVAIDS PDU
Linear Object	Linear Object PDU
Point Object	Point Object State PDU
Propulsion Noise	Underwater Acoustics PDU
Radio Receiver	Radio Receiver PDU
Radio Signal	Signal PDU
Radio Transmitter	Radio Transmitter PDU
Remove Entity	Remove Entity PDU

Table A-2: DIS broker translators

Translator	PDU
Set Data	Set Data PDU
Start	Start Resume PDU
Stop	Stop Freeze PDU

A.2.1. DIS Broker Command-Line Options

The DIS broker command-line options override the settings in a connection's configuration file. The DIS broker has the following command-line syntax:

```
disBroker [
   -- | --ignore_rest
   (-a | --applicationNumber) application
   (-A | --destinationAddress) address
   --brokerName name
   --controlQueueName name
   --debuggedTranslators translator
   --disabledTranslators translator
   --disMcastDevice device
   --disSendPort sendPort
   -E | --dumpEnvironment
   -G | --noGui
   -h | --help
   --interactionQueueName name
   (-l | --configFile) filename
   (-L | --brokerLogFileName) logfile_name
   (-n | --notifyLevel) level
   --objectQueueName name
   -O | --oneToOneMapping
   --pluginDirectory directory
   (-P | --disPort) port
   -qtTranslatorLocale filename
   (-s | --siteId) site
   --subnetMask mask
   --showInfoDialog
   (-t | --tcpPort) port
   -v | --version
   (-x | --exerciseId) exercise]
```

where:

`(-- | --ignore_rest)` tells the broker to ignore the arguments following this flag.

`(-a | --applicationNumber)` *`application`* specifies the unique DIS application number. It is used for creating entity IDs for PDUs originating from the DIS broker. Default: 27.

`(-A | --destinationAddress)` *`address`* is the address of outgoing DIS PDUs. If set to 0.0.0.0 the default broadcast address is used. Default: 0.0.0.0.

`--brokerName` *`name`* specifies the name of the broker. The broker name must be unique among all brokers used. Default: Broker.

`--controlQueueName` *`name`* specifies the name of the shared memory control queue.

`--debuggedTranslators` *`translator`* specifies translators for which you want to enable debugging.

`--disabledTranslators` *`translator`* specifies translators that you want to disable. Accepted multiple times.

`--disMcastDevice` *`device`* is the address of the device to use for multicast traffic. (This option is equivalent to the Network Interface Address parameter on the Connection Information dialog box, Connections page.) If set to 0.0.0.0, the default network address is used. Default: 0.0.0.0.

`--disSendPort` *`sendPort`* specifies the port the broker sends PDUs on. This port is only to be used by the broker. It knows that if it receives a message with this port, that it is its own message, so it does not translate it. Federates listen to the port specified by `--disPort`. Make sure this port does not conflict with others on your network.
Default: 2999.

`(-E | --dumpEnvironment)` print all of the environment variable settings to the log file.

`(-G | --noGui)` specifies that the VR-Exchange Portal GUI should not be run.

`(-h | --help)` displays usage information and exits.

`--interactionQueueName` *`name`* specifies the name of the shared memory interaction queue.

`(-l | --configFile)` *`filename`* specifies the name of the configuration file for the specified broker. Default: *broker.xml*.

`(-L | --brokerLogFileName)` *`filename`* specifies the name of the log file for the broker.

`(-n | --notifyLevel)` *`level`* specifies the output level for the broker's log file. Value values are 0 – 4. As the level increases, so does the output generated. Default: 2. The meanings of the various levels are described in "VR-Exchange Command-Line Options," on page 3-4.

`--objectQueueName` *`name`* specifies the name of the shared memory object queue.

`(-O | --oneToOneMapping)` forces the broker to send an update every time it receives an update from the portal, even if normal rules would dictate that an update is not necessary.

`--pluginDirectory` *`directory`* specifies the location of plug-ins.

`(-P | --disPort)` *`port`* specifies the port to send and receive network PDUs on. Default: 3000.

`--qtTranslatorLocale` *`filename`* specifies the location of the Qt translation file.

`(-s | --siteId)` *`site`* specifies the site ID for the current exercise. This needs to be the same for all participants in the DIS exercise. Default: 1.

`--subnetmask` *`mask`* specifies the subnet mask. VR-Exchange can usually figure out the subnet mask from the IP address. However, if it does not do this correctly, you can explicitly specify it.

`--showInfoDialog` specifies that the connection information dialog box be displayed at startup.

`(-t | --tcpPort)` *`port`* specifies the TCP port to use to maintain connectivity with the Portal. If the Portal dies, then the broker will know by testing this port. If this argument is set to 0, the broker does not try to connect. Default: 5111.

`(-v | --version)` displays version information and exits.

`(-x | --exerciseId)` *`exercise`* specifies the exercise ID for the current exercise. This needs to be the same for all participants in the DIS exercise. Default: 1.

A.3. The DDS Broker

The DDS Broker uses a custom object model that supports entity objects and the fire and detonate interactions. The code for this broker is provided, and can be used to customize the object model or add support for additional objects and interactions.

A.3.1. DDS Translators

Table A-3 lists the DDS translators provided by the DDS broker.

Table A-3: DDS translators

Translator	Class
Entity State	simpleOM::simpleBaseEntity
Fire	simpleOM::simpleFireInteraction
Detonation	simpleOM::simpleDetonationInteraction

A.3.2. DDS Broker Command-Line Options

The DDS broker command-line options override the settings in a connection's configuration file. The DDS broker has the following command-line syntax:

```
ddsBroker [
   --
   --brokerName name
   --controlQueueName name
   --debuggedTranslators translator
   --disabledTranslators translator
   -E | --dumpEnvironment
   -G | --noGui
   -h
   --interactionQueueName name
   (-l | --configFile) filename
   (-L | --brokerLogFileName) logfile_name
   (-n | --notifyLevel) level
   --objectQueueName name
   -O | --oneToOneMapping
   --partitionName name
   --DomainID id
   --pluginDirectory directory
   -qtTranslatorLocale filename
   --showInfoDialog
   (-t | --tcpPort) port
   -v
   -x exercise]
```

where:

`(-- | --ignore_rest)` tells the broker to ignore the arguments following this flag.

`--brokerName` *`name`* specifies the name of the broker. The broker name must be unique among all brokers used. Default: Broker.

`--controlQueueName` *`name`* specifies the name of the shared memory control queue.

`--debuggedTranslators` *`translator`* specifies translators for which you want to enable debugging.

`--disabledTranslators` *`translator`* specifies translators that you want to disable. Accepted multiple times.

`(-E | --dumpEnvironment)` print all of the environment variable settings to the log file.

`(-G | --noGui)` specifies that the VR-Exchange Portal GUI should not be run.

`(-h | --help)` displays usage information and exits.

`--interactionQueueName` *`name`* specifies the name of the shared memory interaction queue.

`(-l | --configFile)` *`filename`* specifies the name of the configuration file for the specified broker. Default: *broker.xml*.

`(-L | --brokerLogFileName)` *`filename`* specifies the name of the log file for the broker.

`(-n | --notifyLevel)` *`level`* specifies the output level for the broker's log file. Value values are 0 – 4. As the level increases, so does the output generated. Default: 2. The meanings of the various levels are described in "VR-Exchange Command-Line Options," on page 3-4.

`--objectQueueName` *`name`* specifies the name of the shared memory object queue.

`(-O | --oneToOneMapping)` forces the broker to send an update every time it receives an update from the portal, even if normal rules would dictate that an update is not necessary.

`--partitionName` *`name`* specifies the DDS partition to join using the OpenSplice Run Time System (RTS). The partition name is similar to a federation in HLA. It groups together DDS applications so that they can see each other on the network.

`--DomainID` is an integer that represents an address space for DDS applications. DDS applications must join the same DDS Domain to communicate.

`--pluginDirectory` *`directory`* specifies the location of plug-ins.

`--qtTranslatorLocale` *`filename`* specifies the location of the Qt translation file.

`--showInfoDialog` specifies that the connection information dialog box be displayed at startup.

`(-t | --tcpPort)` *`port`* specifies the TCP port to use to maintain connectivity with the Portal. If the Portal dies, then the broker will know by testing this port. If this argument is set to 0, the broker does not try to connect. Default: 5111.

`(-v | --version)` displays version information and exits.

A.4. The WebLVC Broker

The WebLVC broker provides comprehensive translation of the WebLVC protocol with its standard object model as well as extensions that support VR-Forces, VR-Vantage, and MAK Logger remote commands. The WebLVC broker has the following translators:

- MAK:VRFBackEnd
- MAK:VRFMessage
- MAK:VRVMessage
- MAK:LoggerControl
- OMT:Interactions
- OMT:Objects
- WebLVC:AggregateEntity
- WebLVC:Data
- WebLVC:EnvironmentalEntity
- WebLVC:MunitionDetonation
- WebLVC:PhysicalEntity
- WebLVC:RadioSignal
- WebLVC:RadioTransmitter
- WebLVC:StartResume
- WebLVC:StopFreeze
- WebLVC:TransferOwnership
- WebLVC:WeaponFire.

A.4.1. WebLVC Broker Command-Line Options

The WebLVC broker command-line options override the settings in a connection's configuration file. The WebLVC broker has the following command-line syntax:

```
disBroker [
   -- | --ignore_rest
   --brokerName name
   --controlQueueName name
   --debuggedTranslators translator
   --disablePublishTranslators translator
   --disableReflectTranslators translator
   -E | --dumpEnvironment
   -G | --noGui
   -h | --help
   --httpProxy
   --httpServer
   --interactionQueueName name
   (-l | --configFile) filename
   --localHttpDirectory directory
   --localNonProxyDir directory
   (-L | --brokerLogFileName) logfile_name
```

```
(-n | --notifyLevel) level
--objectQueueName name
--omtFiles name
-O | --oneToOneMapping
--pluginDirectory directory
--proxyDest hostname
--proxyPort port
-qtTranslatorLocale filename
--sslCertificate name
--showInfoDialog
--sleepInterval interval
--statusPage
(-S | --tcpServerPort) port
--useTcpPort
-v | --version
(-w | --webPort) port]
```

where:

`(-- | --ignore_rest)` tells the broker to ignore the arguments following this flag.

`--brokerName` *`name`* specifies the name of the broker. The broker name must be unique among all brokers used. Default: Broker.

`--controlQueueName` *`name`* specifies the name of the shared memory control queue.

`--debuggedTranslators` *`translator`* specifies translators for which you want to enable debugging.

`--disableReflectTranslators` *`translator`* specifies reflected translators to disable (accepted multiple times).

`--disablePublishTranslators` *`translator`* specifies published translators to disable (accepted multiple times).

`(-E | --dumpEnvironment)` print all of the environment variable settings to the log file.

`(-G | --noGui)` specifies that the VR-Exchange Portal GUI should not be run.

`(-h | --help)` displays usage information and exits.

`--httpProxy`. Enables the HTTP proxy.

`--httpServer`. Enables the web server.

`--interactionQueueName` *`name`* specifies the name of the shared memory interaction queue.

`(-l | --configFile)` *`filename`* specifies the name of the configuration file for the specified broker. Default: *broker.xml*.

`--localHttpDirectory` *`directory`* specifies the local directory where javascript and HTML files for the web server are located.

`--localNonProxyDir` *directory*. If the web server and the proxy are both enabled, this client-side directory signals for pages to be served without using the proxy.

`(-L | --brokerLogFileName)` *filename* specifies the name of the log file for the broker.

`(-n | --notifyLevel)` *level* specifies the output level for the broker's log file. Value values are 0 – 4. As the level increases, so does the output generated. Default: 2. The meanings of the various levels are described in "VR-Exchange Command-Line Options," on page 3-4.

`--objectQueueName` *name* specifies the name of the shared memory object queue.

`--omtFiles` *filename* specifies the OMT or FED files for automatic WebLVC generation.

`(-O | --oneToOneMapping)` forces the broker to send an update every time it receives an update from the portal, even if normal rules would dictate that an update is not necessary.

`--pluginDirectory` *directory* specifies the location of plug-ins.

`--proxyDest` *hostname* specifies the destination host for proxied pages.

`--proxyPort` *port* specifies the destination port for proxied pages.

`--qtTranslatorLocale` *filename* specifies the location of the Qt translation file.

`--showInfoDialog` specifies that the connection information dialog box be displayed at startup.

`--sleepInterval` *interval* specifies the amount of time to wait between every tick. This is not the amount of time between updates, but the amount of time that the broker checks if there is an update to be processed.

`--sslCertificate` *certificate* specifies the path to the file or directory with the CA root certificates. If empty, use HTTP.

`--statusPage` enables the status page that has information about current connections. URL: HTTP://<yourserver>/status.

(`-S | --tcpServerPort`)*port* specifies the port for the alternate TCP server.

`--useTcpPort` Enables an alternate TCP server in addition to the HTML server. This is necessary only if you are sending WebLVC data wihtout using web clients.

`(-v | --version)` displays version information and exits.

(`-w | --webPort`) *port* specifies the HTTP web server port.

MAK Products Glossary

This glossary defines terms used in MAK product documentation.

absolute dead reckoning

A method of dead-reckoning in which an application uses the time stamps contained within state updates if the sender is using absolute time stamping. Contrast with relative dead reckoning.

aggregate

The combination of individual entities, aggregates, or both into a single object. For example, an organizational group such as a platoon.

API

Application Programming Interface.

articulated part

A part of an entity that is capable of movement relative to the entity, such as a turret or gun.

attached part

An object that is attached to another object, such as a rocket attached to a jet.

body coordinate frame

A coordinate framework centered on an entity. To represent the orientation of an entity, an application uses Euler angles or a rotation matrix that rotates from a reference frame to the body coordinate frame.

channel

A channel renders the view of an observer in a 3D visualization application.

clipping

A feature that prevents the display of terrain and objects or parts of objects, that are closer than or farther than specific distances from the observer.

clipping planes

The range of distances from the observer (near clipping and far clipping) in which an application clips objects.

culling

The process of discarding database objects which are not within view, and therefore need not be rendered and displayed.

Cartesian coordinates

A system for indicating location by means of three planes intersecting at right angles to one another at the origin.

Data Distribution Management

The set of HLA services used to specify the routing of data between publishers and subscribers. DDM adds greater control over the Declaration Management (DM) services that only specify filtering based on the class of the data (e.g., ground vehicle versus aircraft). DDM uses the data content to express regions of interest and regions of affect whose overlap establish a communication channel. An example is geographic filtering where a publisher indicates a region around the entity or effect and a subscriber indicates a region expressing its sensor range.

dead-reckoning

A process by which an application calculates the expected location of an entity during periods between state updates, based on velocity, acceleration, and rotational velocity.

DDM

Data Distribution Management.

Defense Modeling and Simulation Office

The former name of the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Renamed Modeling and Simulation Coordination Office. Currently the MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness.

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

An object in an application that can render 3D data.

Distributed Interactive Simulation

The DIS protocol is a set of standards that govern how participating applications share information about the virtual world. The protocol specifies a set of packets, called protocol data units (PDUs), that communicate this information. Each PDU identifies the sender and contains other information depending on the PDU type. The DIS protocol also specifies when and how frequently PDUs are sent.

The DIS protocol has been superseded by the High-Level Architecture for use by the Department of Defense.

DMSO

See Defense Modeling and Simulation Office.

element definition

In VR-Forces and VR-Vantage, an element definition specifies all of the visualizers that control the various aspects of visualizing a scene element. For example, an element definition includes the main model or military symbology, trailing effects, cockpit displays, and so on.

emitter

A simulated device on an entity that emits electromagnetic radiation, for example, radar.

entity

An element in a simulation, such as a vehicle or a person, that is represented in the simulation through the issuance of state messages.

entity maintenance

A process by which the MAK Data Logger compensates for discontinuities following a time jump. The Logger sends out interim state messages for entities that were present before the time jump so that they do not time out before the next update message arrives.

entity-type

A list of seven components as specified by the DIS protocol and the HLA RPR FOM. If none of the seven components contains a wildcard (-1) value, then the entity type refers to a specific, narrowly-defined entity. If some components contain a wildcard, then the entity type refers to a class of entities. A larger number of wildcards indicates a broader, more general class.

The components of an entity-type are:

- Entity kind
- Domain
- Country
- Category
- Subcategory
- Specific
- Extra.

environmental process

According to the IEEE 1278.1a specification (DIS), environmental process PDUs communicate simple environment variables, small scale environmental updates, and embedded processes.

Euler angles

A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). These angles specify successive rotations needed to transform from a reference coordinate system to the entity's body coordinate system.

event

An interaction between objects or between an object and the terrain, such as firing of a munition, or a collision of entities.

exercise

The DIS term for a one or more interacting simulation applications. Compare to federation execution in HLA.

exercise connection

The object (*DtExerciseConn*) through which a VR-Link application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection. (All MAK products are VR-Link applications.)

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

Federation Execution Data file. The Federation Execution Data is the subset of FOM data needed and read by the RTI. VR-Link applications also read the FED file.

federate

A connection to the RTI. Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same federation execution.

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Controls the perspective of the observer.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the observer, since the observer coverage spans a wider area. Depth become exaggerated.

A narrow field of view creates an effect like that of a telephoto lens. Objects appear larger and closer to the observer, and the overall scene depth appears flattened. The distances between objects appears compressed.

filter range

A setting that prevents distant entities from being processed while allowing distant terrain to appear normally.

FOM

See Federation Object Model.

FOM Module

In HLA Evolved, defines additional modular data content for a federation execution, usually extensions to an existing FOM. (Also supported by the HLA 1.3 and HLA 1516 versions of the MAK RTI. Per the HLA Evolved interface specification:

"A partial FOM (containing some or all OMT tables) that specifies a modular component of a FOM. A FOM module may contain classes not inherent to it but upon which the FOM module depends, i.e., superclasses to the modular components. These superclasses will be included in the FOM module either as complete or scaffolding definitions."

frame rate

The rate at which the application displays updated images.

ground clamping

A process by which the a 3D visualization application keeps an entity anchored to the surface of its terrain database, regardless of the altitude data contained in its entity state message.

geocentric coordinates

A coordinate system calculated with respect to the earth's center. The origin of the geocentric coordinate system is the center of the earth. The positive X-axis passes through the prime meridian at the equator; the positive Y-axis passes through 90 degrees east longitude at the equator; and the positive Z-axis passes though the north pole.

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MAK applications, geodetic coordinates consist of latitude and longitude in radians, and altitude in meters above the reference ellipsoid.

gridded data

Data that has been processed in a rectangular array of points, in X, Y or latitude/longitude, at which single data values define a two dimensional function. According to the IEEE 1278.1a specification, gridded data transmits information about large-scale or high-fidelity spatially and temporally varying ambient fields and about environmental processes and features.

guise

An alternative entity type used to display an object depending on the force ID. For example, a tank could look like an M1A1 to friendly forces and a T72 to hostile forces.

head-up display

A set of indicators and readouts superimposed onto a graphics display. Also called an overlay.

heartbeat

In DIS, the frequency with which current PDUs are sent to the network regardless of whether or not the entity's state has changed.

High-Level Architecture

The High Level Architecture (HLA) for simulations is a U. S. Department of Defense (DOD)-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the DOD.

HLA

See High-Level Architecture.

interaction

A message describing a simulation event. An interaction describes an event, it does not update an object's state.

Logger control PDUs

A set of DIS PDUs or HLA interactions that let you control the MAK Logger remotely.

logical range

A suite of TENA Resources, sharing a common object model, that work together for a given range event. Similar in concept to the HLA Federation.

Local RTI Component

The libraries on a computer that implement an RTI. A federate communicates with the HLA network through the LRC.

LRC

See Local RTI Component.

MAK Technologies Lisp

An adaptation of the Lisp language used in configuration files for MAK products.

message

A general term used to refer to DIS PDUs and HLA interactions and state updates. In TENA, a message is similar to an HLA interaction.

MIM

The MOM & Initialization Module (MIM) allows for extensions to be made to the HLA standard MOM. It is a subset of the FOM that contains OMT tables that describe the HLA MOM. The MIM also contains additional predefined HLA constructs such as object and interaction roots, data types, transportation types and dimensions. HLA specifies a standard MIM that is incorporated into all FDDs automatically by the RTI. The standard MIM can be replaced with a user-supplied MIM containing the standard MIM plus extensions.

model definition

In VR-Forces and VR-Vantage, a model definition specifies the 3D model and other basic attributes of a model. It is referenced by an element definition.

Modeling and Simulation Coordination Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provided a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. The MSCO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness. (Formerly DMSO)

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MAK Technologies Lisp.

orientation clamping

Adjusts the pitch and roll of an entity so that it appears properly seated when the terrain is inclined. For example, if a tank is moving horizontally across the face of a hill, orientation clamping prevents the tank from appearing level, and therefore, partially embedded into the hillside. Used with ground clamping.

object

An element in a simulation that has persistence, as opposed to an interaction, which is a transient element.

object handle

An integer that an application uses to identify a particular object in RTI service calls. An object handle is meaningful only to a particular federate. The same object can be known to different federates by different object handles.

object name

A character string that can be used to identify an object. In HLA, the object name is known to the RTI, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

observer

In MAK 3D applications, the point of view into the scene. (Called the eyepoint in MAK Stealth 6.x and earlier.)

PDU

See Protocol Data Unit.

Protocol Data Unit

A unit of data message (packet) that is passed on a network between DIS simulation applications.

radio transmitter

A simulated device on an entity, capable of transmitting radio communications.

radio receiver

A simulated device on an entity, capable of receiving radio communications.

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

A Data Logger file that stores a history of the interactions in a simulation for playback.

reflected entity list

A list of entities simulated by remote applications (federates in HLA) and about which the local simulation has received information over the network.

reference FOM

A FOM designed to be used as a whole, or with modification, by a wide variety of similar federation executions.

relative dead reckoning

A method of dead reckoning in which an application approximates the location of an object based on the local time of receipt of the last state update message.

remote display engine

A display engine running in an application that is separate from the master application, and which is controlled by the master application.

RID file

See RTI Initialization Data.

RTI Initialization Data

The initialization data required by the RTI for operation.

Data required by an RTI during initialization, independent of the FOM being used. RID data is usually dependent on a specific implementation of the RTI.

RPR FOM

See Realtime Platform Reference FOM.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

A library and other supporting software that implements the HLA interface specification. All federates communicate with one another in an HLA environment through RTI functions.

scene element

In VR-Vantage and VR-Forces, any object that is rendered in the scene, such as an entity model, interaction, or tactical graphic.

Simulation Interoperability Standards Organization

A group that seeks to promote modeling and simulation interoperability and reuse for the benefit of diverse M&S communities, including developers, procurers, and users, world-wide.

simulation time

In VR-Forces, simulation time is used in dead-reckoning of remote entities and thresholding of local entities. Typically, simulation time is set once during each iteration of the application's main simulation loop so that all entities are dead-reckoned based on the same value of current time.

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position are not so abrupt as to be visually disconcerting.

state

The current status of an object, including location, direction of movement, extent of damage, and so on.

tape

An alternative term for a Logger recording. Not a physical magnetic tape.

terrain following

In a 3D visualization application, causes the observer (eyepoint) to maintain a constant distance above the terrain surface. The observer's height changes in tandem with the peaks and valleys as it passes over the geography.

timeout

The period of time in which an application continues to display an entity after that entity's update messages have stopped appearing on the network. Time-outs are usually not used in HLA because there is no set frequency (no heart-beat) for transmitting messages.

timescale

A factor by which time is accelerated or slowed during playback of a Data Logger recording.

topographic coordinates

A right-handed Cartesian coordinate system whose X-Y plane is tangent to the earth's surface at the origin, with the positive X axis pointing north, the positive Y axis pointing east, and the positive Z axis pointing down. There are an infinite number of topographic coordinate systems – one for each point on the earth's surface.

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

A method used by to smooth positional discontinuities that could occur when new state updates arrive.

UDP port

A network channel through which an application sends and receives data for DIS exercises.

Universal Transverse Mercator

In general, a non-cartesian coordinate system in which the X, Y, and Z correspond to easting (nearly east), northing (nearly north), and height above an ellipsoid that approximates the surface of the earth.

UTM

See Universal Transverse Mercator.

view control messages

A set of programmatic messages that let you control the view in VR-Vantage applications remotely.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X-Y-Z

Link - Simulate - Visualize

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