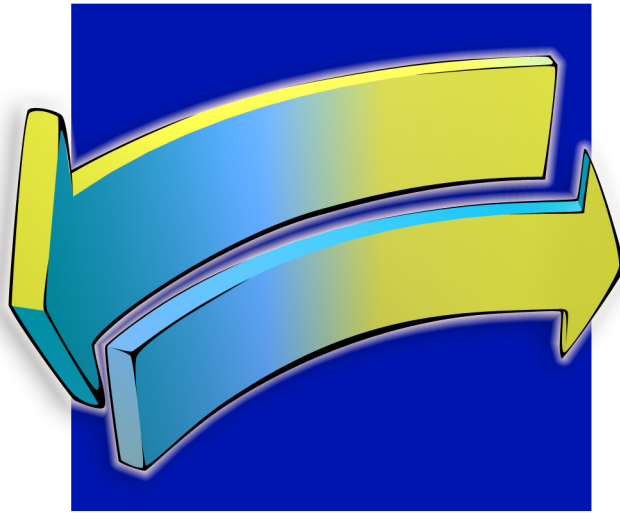




User's Guide





MÄK Gateway

User's Guide

Copyright © 2005 MÄK Technologies
All rights Reserved. Printed in the United States.

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

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

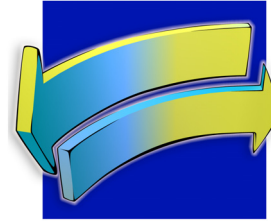
All other trademarks are owned by their respective companies.

MÄK Technologies
10 Fawcett St.
Cambridge, MA 02138 USA

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

info@mak.com
www.mak.com

Revisions GWY-4.1.4-1-050811



Contents

Preface

How this Manual is Organized	vii
MÄK Products	viii
How to Contact Us	x
Document Conventions	xi
Hardware and Operating System Naming Conventions.....	xi
Mouse Button Naming Conventions	xii

Chapter 1 Introduction

1.1. Overview	1-2
1.2. Translating To and From HLA	1-3
1.2.1. RTI Support	1-3
1.2.2. FOM Support	1-3
1.2.3. Handle Mapping	1-4
1.2.4. Attributes Required for Sending DIS PDUs	1-5
1.3. Translating Between DIS and the RPR FOM	1-6
1.3.1. Entity Information and Entity Interaction	1-7
1.3.2. Warfare	1-8
1.3.3. Simulation Management	1-9
1.3.4. Radio Communications	1-10
1.3.5. Distributed Emission Regeneration	1-11
1.3.6. Logistics	1-12
1.3.7. Entity Management	1-12
1.3.8. Synthetic Environment	1-12
1.3.9. MÄK Logger Control and MÄK Stealth View Control PDUs	1-12

Chapter 2 Installing the Gateway

2.1. Installing MÄK Gateway	2-2
2.1.1. Installing MÄK Gateway on Windows	2-2
2.1.2. Installing MÄK Gateway on a UNIX System	2-3
2.1.3. Installing the Class Reference Files	2-4
2.2. Directory Structure	2-5
2.3. Setting Up and Using the FLEXlm License Manager	2-6
2.3.1. The FLEXlm Client-Server Architecture	2-6
2.3.2. Installing and Configuring the License Manager	2-7
2.3.3. Installing the License Manager Software	2-7
2.3.4. Requesting a License File	2-8
2.3.5. Putting the License File in the <i>flexlm</i> Directory	2-9
2.3.6. Setting the MAKLMGRD_LICENSE_FILE Variable	2-9
2.3.7. Adding Additional License Files	2-10
2.3.8. Running the License Server	2-11
2.3.9. Running FLEXlm as a Service on Windows	2-12
2.3.10. Troubleshooting the FLEXlm License Manager	2-13
2.4. Installing an RTI	2-15
2.4.1. Installing the MÄK RTI	2-15
2.4.2. Installing the DMSO RTI	2-16

Chapter 3 Using the Gateway

3.1. Overview	3-3
3.2. Setting Up the RTI Environment	3-3
3.3. Starting the Gateway	3-3
3.3.1. Starting the Gateway in UNIX	3-3
3.3.2. Starting the Gateway on a PC	3-4
3.3.3. Starting Message Translation Immediately After Startup	3-5
3.3.4. Gateway Startup Options	3-5
3.3.5. Closing the Gateway	3-6
3.4. Configuring the Gateway	3-6
3.4.1. Configuring the Gateway in the GUI	3-7
3.4.2. Specifying the FOM Mapper and FED File to Use	3-8
3.4.3. Picking the Ethernet Card to Use	3-9
3.5. Translating DIS and HLA Messages	3-10
3.5.1. Viewing Objects and Interactions	3-10
3.5.2. Enabling and Disabling the Display of Object Data	3-11
3.5.3. Absolute Timestamp Mode	3-12
3.5.4. Forcing Updates	3-12
3.5.5. Stopping Message Translation	3-12

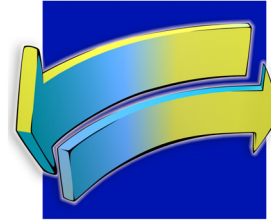
3.6. Choosing the Objects and Interactions to Translate	3-13
3.6.1. Filtering the Display by Object or Interaction Type	3-13
3.6.2. Disabling Translators by Default	3-14
3.6.3. Filtering Translation of Individual Objects	3-14
3.6.4. Translating Incomplete Messages	3-16
3.6.5. Parsing Unknown Global IDs	3-16
3.6.6. Specifying the Number of Received Interactions to Display	3-16
3.7. Debugging Translations	3-17
3.7.1. Turning on Debugging by Default	3-17
3.7.2. Dumping Text to a File	3-18
3.8. Auto Restart	3-18

Appendix A Gateway MTL Parameters

A.1. MÄK Technologies Lisp (MTL) Files	A-2
A.1.1. Using Environment Variables in an MTL File	A-2
A.1.2. Specifying Lists of Lists	A-3
A.1.3. Using Conditional Statements	A-3
A.2. Gateway Configuration Parameters	A-4

MÄK Products Glossary

Index



Preface

The MÄK Gateway allows DIS simulations to interoperate with HLA federations that use the Real-Time Platform Level Reference (RPR) FOM (and other FOMs using VR-Link FOM mappers).

How this Manual is Organized

This manual is written for persons who will install or use the MÄK Gateway. The manual assumes that you are familiar with the basic operation of your operating system and that you know how to work in your computer's graphical window environment.

The chapters are organized as follows:

Chapter 1, *Introduction*, briefly describes the features and uses of the Gateway.

Chapter 2, *Installing the Gateway*, explains how to install the Gateway and other necessary software, including license management software and an RTI.

Chapter 3, *Using the Gateway*, explains how to start the Gateway, configure it, and translate state update messages.

Appendix A, *Gateway MTL Parameters* describes MTL file syntax and the parameters used in the *gateway.mtl* file.

MÄK Products

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

- ♦ VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the 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.

VR-Link is the basis of the other MÄK products.

- ♦ MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK RTI is optimized for high performance. It has an API, RTIsPy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface for the MÄK RTI rtiexec.
- ♦ VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a graphical user interface, that gives you a two dimensional view of a terrain database.

You can create and view local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.

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

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

- ♦ MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal, or in reverse, and locate areas of interest quickly. The Logger has a graphical user interface and a command line interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application.

The Logger comes with a set of utility programs that let you inspect, combine, and edit Logger recordings.

- ♦ MÄK Plan View Display. The Plan View Display (PVD) provides a two-dimensional, “big picture” of a virtual environment. It shows simulated entities, such as vehicles, soldiers, and tanks moving over a 2D-map display.
- ♦ VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate.
- ♦ MÄK Gateway. The Gateway is a translator application that allows DIS exercises to operate as members of HLA federation executions that use the RPR FOM, thereby preserving the value of simulation code written to support DIS. The MÄK Gateway is a standalone application and does not require you to write additional code for its use.
- ♦ MÄK HLA/DIS Toolbox for MATLAB and Simulink. The Gateway is a wrapper around VR-Link that allows MATLAB and Simulink users to analyze data from live or recorded HLA or DIS exercises. It also allows Simulink users to build models that can participate in HLA or DIS exercises.
- ♦ MÄK Game-Link is a toolkit that allows games developed using Epic Games’ Unreal Engine to interoperate with DIS and HLA simulations and trainers.

How to Contact Us

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

Telephone

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

E-mail

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

Internet

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

Post

Send postal correspondence to:	MÄK Technologies 10 Fawcett St. Cambridge, MA, USA 02138
--------------------------------	--

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

Document Conventions

This manual uses the following typographic conventions:

Monospaced	Indicates commands or values you enter.
<i>Monospaced Italic</i>	Indicates command variables that you replace with appropriate values.
{ }	Indicates required arguments.
[]	Indicates optional arguments.
	Separates options in a command where only one option may be chosen at a time.
/	Indicates a directory. Since MÄK products run on both UNIX and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
<i>Italic</i>	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose File → Save.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

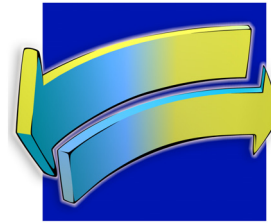
Directory names are preceded with dot and slash characters that show their position with respect to the Gateway home directory. For example, the directory *gateway/doc* appears in the text as *./doc*.

Hardware and Operating System Naming Conventions

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

Mouse Button Naming Conventions

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



1

Introduction

This chapter introduces you to the Gateway and its features.

Overview	1-2
Translating To and From HLA.....	1-3
RTI Support.....	1-3
FOM Support	1-3
Handle Mapping	1-4
Attributes Required for Sending DIS PDUs	1-5
Translating Between DIS and the RPR FOM.....	1-6
Entity Information and Entity Interaction.....	1-7
Warfare.....	1-8
Simulation Management	1-9
Radio Communications	1-10
Distributed Emission Regeneration	1-11
Logistics	1-12
Entity Management.....	1-12
Synthetic Environment.....	1-12

1.1. Overview

The MÄK Gateway provides a means for DIS applications to interoperate with HLA federations that use the Real-Time Platform Level Reference (RPR) FOM. The Gateway is a stand-alone translator application that connects DIS exercises to an HLA RPR FOM federation execution. The Gateway acts simultaneously as a member of the DIS exercise, and together with one or more DIS applications, as a federate in the HLA federation execution (Figure 1-1). Because the Gateway is a stand-alone interface, no modification of the DIS applications is required.

The Gateway is based on the VR-Link Toolkit. Therefore, it can take advantage of VR-Link's FOM agility feature. This means that if you write a FOM Mapper, you can use the Gateway with FOMs other than the RPR FOM. See the VR-Link documentation for details.

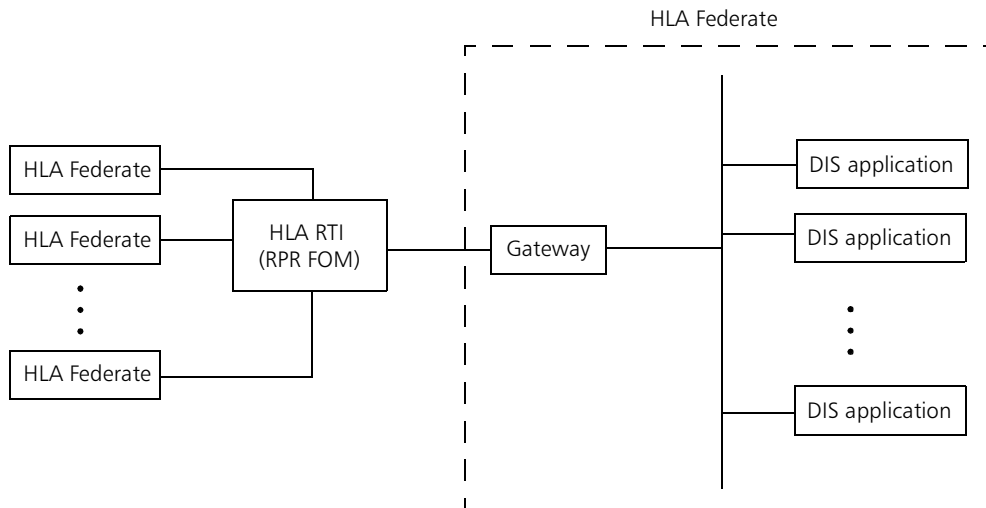


Figure 1-1. Gateway interoperability

DIS to HLA translation works as follows:

1. On the DIS side of the Gateway, data packets called Protocol Data Units (PDUs) are formatted, sent, and received.
2. The Gateway receives the PDUs.
3. It puts them into a VR-Link state repository.
4. The state repository is copied to an HLA state repository.
5. The FOM mapper converts the data in the state repository into and encodes HLA objects and attributes.
6. The HLA data is sent over the network.

In order to communicate with an HLA federation execution, the Gateway must perform some HLA services that have no corresponding function in DIS. These functions include creating, destroying, joining, and resigning federations and publishing and subscribing to RPR FOM classes.

A detailed description of how DIS PDUs are translated into the RPR FOM object and interaction classes is in [“Translating Between DIS and the RPR FOM,”](#) on page 1-6 and Guidance Rational Interoperability Modalities for the Real-Time Platform Reference (GRIM RPR) FOM, Simulation Interoperability Standards Organization (SISO) RPR FOM SDG, <http://www.sisostds.org/stdsdev/rpr-fom/index.htm>.

1.2. Translating To and From HLA

This section discusses general issues involved in translating update messages to and from HLA.

1.2.1. RTI Support

The Gateway supports the HLA RTI 1.3 specification and the SISO Dynamic Link Compatible (DLC) HLA API Standard for the HLA Interface Specification (SISO-STD-004.1-2004). The SISO DLC 1516 API is an alternative mapping of the specification to C++. It supports dynamic link compatibility between different RTI implementations. If you use the 1516 version of the Gateway, all HLA objects created by the Gateway are given a name chosen by the RTI, not the Gateway.

1.2.2. FOM Support

As noted previously, the Gateway provides native support for several versions of the RPR FOM. (See the *MÄK Gateway Release Notes* for a list of supported versions.) That is, the Gateway can translate data between DIS and supported versions of the RPR FOM without any additional code. Like other MÄK products, the Gateway comes with a FED file called *VR-Link.fed*, which contains all of the RPR FOM 1.0 objects and interactions, plus a few extra classes described in the file's comments. By default, the Gateway joins the federation execution called VR-Link, and uses this FED file, which is distributed with the Gateway, and with other MÄK tools. For details about choosing the version of the RPR FOM to use, see [“Specifying the FOM Mapper and FED File to Use,”](#) on page 3-8.

Using FOMs other than the RPR FOM

If you want to use the Gateway with FOMs other than the RPR FOM, you can write a VR-Link FOM mapper to provide the information necessary to properly map between VR-Link's internal object model and your FOM. See *VR-Link Developer's Guide* for complete details about creating FOM mappers.

The Gateway SOM

When a DIS application uses the Gateway to participate in an HLA federation execution, the DIS application and Gateway taken together comprise a federate. Like any other federate, this federate has a SOM that must be submitted when the federate undergoes HLA compliance testing. The contents of the SOM depend on which elements of the FOM are needed by the DIS application. For example, if the DIS application does not care about Collision PDUs, you can instruct the Gateway to not subscribe to (and thus translate) Collision interactions, meaning that Collision interactions will not be part of your federate's SOM.

1.2.3. Handle Mapping

The RTI identifies classes, attributes, and parameters using handles. A federate discovers these handles by submitting the class, attribute, or parameter string name for which it needs a handle. The RTI returns the handle. For efficiency reasons, the Gateway performs this mapping between string names and handles only once and stores the mapping information in an internal database.

Handle mapping is performed immediately prior to publication or subscription. So, the Gateway performs handle mapping only for those classes that are published or subscribed. Therefore, the Gateway can operate with a FED file that contains only those classes that it publishes and subscribes. The Gateway can also tolerate missing attributes or parameters.

During a handle mapping operation, the Gateway attempts to map the complete set of attributes or parameters for each published or subscribed class. If an attribute or parameter is missing, the Gateway gets the default value specified by the VR-Link state repository, usually 0 or type unknown.

1.2.4. Attributes Required for Sending DIS PDUs

HLA object updates contain only the information that has changed since the previous update. Therefore, when the Gateway discovers an HLA object, it is possible that the update message will not include all of the information necessary to complete a PDU. The Gateway will not send the PDU for this object until it has all of the information it needs. [Table 1-1](#) lists the required attributes, the containing class, and comments on any indirect requirements.

Table 1-1: Mandatory Attributes

Class	Attribute / Parameter	Comment
<i>EmitterSystem</i>	EmitterName, EmitterNumber, HostObjectID, (active beams)	HostObjectID (emitting entity) must refer to an object that has been registered or discovered by the Gateway. The system must have active beams except when being removed.
<i>EmitterBeam</i>	BeamID, EmittingSystemID, BeamParameterIndex, ERP	EmittingSystemID must refer to an object that has been registered or discovered by the Gateway. Heartbeat updates require BeamParameterIndex and ERP to be non-zero.
<i>EntityState</i>	EntityIdentifier	An EntityIdentifier is valid if it is not 0:0:0. This attribute is needed only if renameDisObjects is set to 0.
<i>Designator</i>	HostObjectID	HostObjectID (designating entity) must refer to an object that has been registered or discovered by the Gateway.
<i>RadioTransmitter</i>	HostObjectID	HostObjectID (transmitting entity) must refer to an object that has been registered or discovered by the Gateway.
<i>RadioReceiver</i>	HostObjectID	HostObjectID (receiving entity) must refer to an object that has been registered or discovered by the Gateway.
<i>IFF</i>	HostObjectID, SystemType	HostObjectID (emitting/receiving entity) must refer to an object that has been registered or discovered by the Gateway.

1.3. Translating Between DIS and the RPR FOM

The Gateway handles the discrepancy between the DIS concept of a heartbeat and the HLA's lack of such as follows:

- ♦ The Gateway provides heartbeat updates for the appropriate PDUs when the time between HLA updates exceeds the heartbeat threshold.
- ♦ The Gateway deletes HLA objects when the time between DIS PDUs exceeds the timeout threshold ($2.5 \times \text{heartbeat}$). RPR FOM embedded system objects (radio transmitter/receiver and emitter system/beams) are deleted only if their host entity no longer exists.

The Gateway does not support the full set of PDUs in the 1278.1 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
 - Environmental Process PDU
 - Gridded Data PDU.
- ♦ Logger Control PDUs (for MÄK Logger)
- ♦ View Control PDUs (for MÄK Stealth).

The following sections describe the supported PDUs in greater detail.

1.3.1. 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.

1.3.2. 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.

1.3.3. 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

1.3.4. 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 the Gateway. 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.

Table 1-5: RadioSignal Interaction Class

Class1	Class2
RadioSignal	ApplicationSpecificRadioSignal
	DatabaseIndexRadioSignal
	EncodedAudioRadioSignal
	RawBinaryRadioSignal

1.3.5. 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 the Gateway 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 *Emitting-SystemID* that establishes the association between it and its *EmitterSystem* object instance.

The Gateway creates a Delta State EE PDU for each *EmitterSystem* or *EmitterBeam* reflection. The Gateway does not generate a delta EE PDU if the *EmitterSystem* has no active beams (unless that beam is being removed). The Gateway 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. The Gateway 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.

1.3.6. 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.

1.3.7. Entity Management

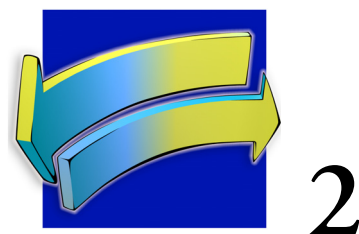
The Transfer Control Request PDU is translated to the HLA *TransferControl* interaction class. The *TransferControl* class is available only if you are using RPR FOM version 2, draft 6 or later.

1.3.8. Synthetic Environment

The Gateway supports two PDUs in the Synthetic Environment family: Environmental Process PDU and Gridded Data PDU. The Environmental Process PDU is translated to the *EnvironmentProcess* class. The Gridded Data PDU is translated to the *Gridded-Data* class.

1.3.9. MÄK Logger Control and MÄK Stealth View Control PDUs

In addition to the standard DIS PDUs described in previous sections, the Gateway supports Logger Control PDUs for controlling the MÄK Logger and View Control PDUs for controlling the MÄK Stealth.



Installing the Gateway

This chapter explains how to install Gateway software, RTI software, and license management software.

Installing MÄK Gateway.....	2-2
Installing MÄK Gateway on Windows	2-2
Installing MÄK Gateway on a UNIX System	2-3
Directory Structure.....	2-5
Setting Up and Using the FLEXlm License Manager	2-6
The FLEXlm Client-Server Architecture	2-6
Installing and Configuring the License Manager.....	2-7
Installing the License Manager Software	2-7
Requesting a License File.....	2-8
Putting the License File in the flexlm Directory.....	2-9
Setting the MAKLMGRD_LICENSE_FILE Variable.....	2-9
Adding Additional License Files.....	2-10
Running the License Server	2-11
Troubleshooting the FLEXlm License Manager	2-13
Installing an RTI.....	2-15
Installing the MÄK RTI	2-15
Installing the DMSO RTI	2-16

2.1. Installing MÄK Gateway

This section explains how to install Gateway on a Windows or a UNIX operating system. You must also install the license manager files (see [“Setting Up and Using the FLEXlm License Manager,”](#) on page 2-6) and if you want to use the Gateway with HLA, an RTI (see [“Installing an RTI,”](#) on page 2-15).

2.1.1. Installing MÄK Gateway on Windows

This section explains how to install Gateway from CD-ROM or from an ftp download on a PC.

Before you install Gateway, read the *Release Notes* to see if there are any special instructions for installation.

Installing MÄK Gateway from a CD

Gateway may be supplied to you on an “All Products” CD with a common installation interface or a CD that does not support common installation.

To install Gateway from a CD:

1. Insert the Gateway CD into your CD-ROM drive.
 - If this is an All Products CD, and the autorun feature is enabled, the MÄK PC Products Installation window opens. The PC Products Installation window is similar to [Figure 2-1](#).

If the MÄK PC Products Installation window does not open, open the Windows Explorer. In the root directory for your CD drive, double-click *makInst.exe*. The MÄK PC Products Installation window opens.
 - If this is not an All Products CD, and autorun is enabled, an installation program may start immediately. Follow the instructions of the installation program.

If the installation does not start automatically, open the Windows Explorer and select the CD drive. In the root directory for Gateway, double-click *setup.exe*. Go to step 3.
2. In the MÄK PC Products Installation window, click Gateway.
3. Follow the instructions of the installation program.

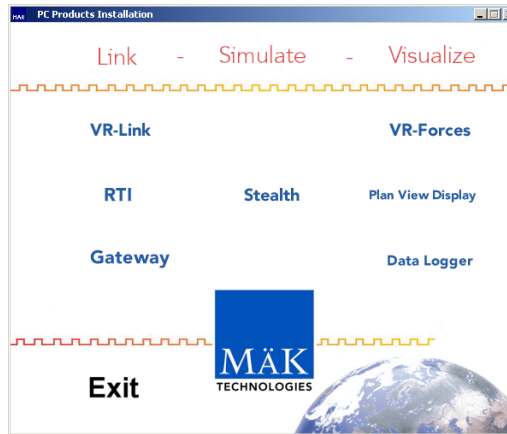


Figure 2-1. MÄK PC Products Installation window

Installing MÄK Gateway from a Downloaded File

If you obtained Gateway via ftp, you received a compressed file. It may be a zip file or an executable file.

To install Gateway from a zip file:

1. Copy the zip file to a temporary directory.
 2. Unzip the zip file.
 3. Run *setup.exe*.
 4. Follow the instructions of the installation program.
- To install Gateway from an executable file, run the executable.

2.1.2. Installing MÄK Gateway on a UNIX System

This section explains how to install Gateway on UNIX systems from CD-ROM or from an ftp download.

Installing MÄK Gateway from a CD-ROM

To install Gateway:

1. Mount the CD-ROM.
2. Run:

```
./install
```
3. Follow the on-screen instructions. You can install Gateway into any directory for which you have write permissions.

Installing MÄK Gateway from a Downloaded File

If you obtained Gateway via ftp, you received a compressed tar file.

To install Gateway:

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

```
gtar xzf gatewayxx-os.tar.gz
```



You must use gtar (from GNU) not the operating system's default tar.

2.1.3. Installing the Class Reference Files

On Windows, if your product has class reference files, they may be installed as zip files. If this is the case and you want to use the class reference, extract the files into the *./doc/classdoc* directory.

2.2. Directory Structure

The directory structure for the installed files is:

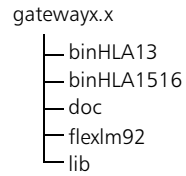


Figure 2-2. Gateway directory structure

[Table 2-1](#) describes the contents of the Gateway directories

Table 2-1: Gateway directory contents

Directory	Contents
<i>binHLA13</i>	Executables, FED files, and configuration files for use with federations using the HLA 1.3 specification.
<i>binHLA1516</i>	Executables, FED files, and configuration files for use with federations using the HLA 1516 specification.
<i>doc</i>	Gateway documentation.
<i>flexlm92</i>	License management files.
<i>lib</i>	Library files.

2.3. Setting Up and Using the FLEXlm License Manager

MÄK products use the FLEXlm license manager. Before you can use a MÄK product, you must obtain a valid license file and configure the license server and client machines. (For general information about FLEXlm, see <http://www.macrovision.com/products/flexlm/index.shtml>.) Contact the MÄK Technologies sales department for information about special licensing agreements.

2.3.1. The FLEXlm Client-Server Architecture

This section explains the FLEXlm architecture to provide a context for the setup instructions that follow. FLEXlm uses a client-server architecture. The FLEXlm server daemon, *lmgrd*, runs on one computer, known as the license server. This machine services every machine (including possibly itself) on which you want to run a MÄK Technologies product. The machines on which the MÄK Technologies products are running are called clients.

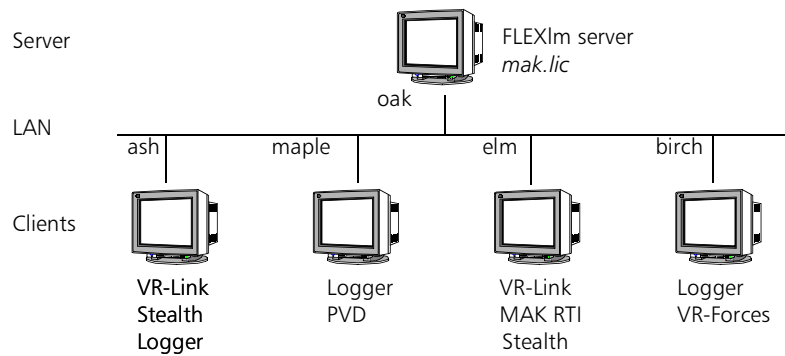
The license server executes vendor daemons (for MÄK Technologies, *maklmgrd*) and uses one or more license files, provided by individual vendors. The license file identifies the products for which you have licenses. On each client machine, the `MAKLMGRD_LICENSE_FILE` environment variable (which you must set) points to the server.

Licenses “float”, so you can install product software on as many computers as you want, but at any given time, you can run only as many instances of a product as you have licensed.



The license key in the license file is tied to the host ID of the server machine. Therefore, if you decide to change the server (or replace the network card), you need to get a new license file. If you change the name of the server, you must update the `MAKLMGRD_LICENSE_FILE` environment variable on every client. Depending on how many clients you have, this may not be a trivial process. Keep this in mind when you choose the machine to use as a license server.

Figure 2-3 illustrates the client-server architecture for FLEXlm and MÄK products.



The MAKLMGRD_LICENSE_FILE environment variable is set to @oak on each client.

Figure 2-3. FLEXlm example architecture

2.3.2. Installing and Configuring the License Manager

The general procedure for installing and configuring license management is as follows (each step is explained in the rest of this section):

1. Install license manager software.
2. Request a license file.
3. Put the license file in the *flexlm* directory on the license server.
4. Set the MAKLMGRD_LICENSE_FILE environment variable on all client machines.

If you have MÄK Technologies products installed and you purchase additional licenses or products, you must get a license file for the new products or licenses. For this procedure, see [“Adding Additional License Files,”](#) on page 2-10.

2.3.3. Installing the License Manager Software

As part of the installation of MÄK products, the FLEXlm software is installed in *./flexlm92*.

1. Install your MÄK products on the machines on which you want to run them.
2. Copy the files in *flexlm92* to *C:\flexlm* on the server machine.
3. Copy the files in *flexlm92* to the server machine. For Windows, copy the files to *C:\flexlm*. For a UNIX computer, a typical location is */usr/local/flexlm*.



Note the difference between the directory name of the supplied files and the directory to which you copy them.

2.3.4. Requesting a License File

When you request a license file, you need to know the:

- ♦ Host ID of the license server
- ♦ Name of the license server
- ♦ MÄK invoice number of your purchase
- ♦ Basic contact information for you and your company
- ♦ Platforms for which you have purchased MÄK products
- ♦ Number of licenses for each platform and product.

When you have this information, go to <http://www.mak.com/licenses.php> and complete the license key request form. MÄK will e-mail you a license file.

Identifying the Host ID and License Server Name in Windows

To find out the Host ID and license server name in Windows:

1. In the flexlm directory, run *lmtools.exe*. The LMTOOLS application opens.
2. Select the System Settings tab. The Host ID is the string in the Ethernet Address text box. The license server name is the name in the Computer/Hostname text box.

Identifying the Host ID and License Server Name on UNIX Computers

To find out your host ID:

1. On the license server, in a command window, change to the *flexlm* directory.
2. Execute *lmhostid*. This program reports a unique number that MÄK uses to generate your license activation codes. Write down the number.



The host ID is keyed to your ethernet card, so if you change your network card, you must request a new license file.

- To find out the name of the server machine, at a command prompt, type `hostname`.

2.3.5. Putting the License File in the *flexlm* Directory

The license file you receive may be an attachment or embedded in an email.

1. If the attachment is not named *mak.lic*, rename it *mak.lic*.
If the license is embedded in an email, copy it into an empty text file. Name the file *mak.lic*.
2. Put the license file in the *flexlm* directory on the license server.



If you already have a *mak.lic* file in the *flexlm* directory, do not overwrite the file. See the instructions in [“Adding Additional License Files,”](#) on page 2-10.

2.3.6. Setting the MAKLMGRD_LICENSE_FILE Variable

The MAKLMGRD_LICENSE_FILE environment variable identifies the server machine so that the FLEXlm software on a client can check out a license when you run a MÄK Technologies product. You must set MAKLMGRD_LICENSE_FILE on every client machine.



If you are running MÄK Technologies products on the license server machine, it is also a client, so you must set MAKLMGRD_LICENSE_FILE on that machine.

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 MÄK Technologies products run on.

Windows 2000

To add the MAKLMGRD_LICENSE_FILE in Windows 2000:

1. Right-click the My Computer icon on the Desktop.
2. On the pop-up menu, select Properties. The System Properties dialog box opens.
3. Click the Advanced tab.
4. Click the Environment Variables button. The Environment Variables dialog box opens.
5. Click the New button. 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.

Windows XP

To add the MAKLMGRD_LICENSE_FILE in Windows XP:

1. On the Start menu, choose My Computer.
2. In the My Computer window, under System Tasks, click View System Information. The System Properties dialog box opens.
3. Follow steps 3 through 8 in the Windows 2000 procedure.

UNIX

On UNIX, 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 MÄK products. See [“Running the License Server,”](#) on page 2-11.

2.3.7. Adding Additional License Files

If you buy additional MÄK products, or additional licenses for products you already have, you must get an additional license file.

- To request an additional license file, follow the same procedure as for your first license file. See [“Requesting a License File,”](#) on page 2-8.

To add a new license file to your license server:

1. When you get the new license file, name it *mak2.lic*, or some other name ending in *.lic*, so that it has a different name from the license files currently in the *flexlm* directory on the license server.
2. Put the renamed file in the *flexlm* directory on the license server, with the other license files.
3. Restart the license server.

You can now use the new products or additional product licenses.

2.3.8. Running the License Server

The FLEXlm license server must be running on the server machine before you can run your MÄK applications.

- To start the license server, on the server machine only, execute `runLm` from the `./flexlm` directory.



Do not execute more than one instance of `runLm` at a time. If you aren't sure whether or not the license server is running, execute `lmstat`.

-
- To obtain the current status of the server, run `lmstat`.

Shutting Down the License Server

- To force all applications to check in their licenses and shut down the license server, run `lmdown`.

2.3.9. Running FLEXlm as a Service on Windows

You can configure FLEXlm to run as a service on Windows 2000 or XP.

To configure FLEXlm as a service:

1. Install the license management software and obtain a license as described previously in this chapter.
2. Run *C:\flexlm\lmtools.exe*.
3. On the Service/License File tab, select the Configuration Using Services option.
4. Select the Config Services tab.
5. In the Service Name box, enter a name for the service.
6. In the Path To The lmgrd.exe File box, enter the path of the directory where the *lmgrd.exe* file is located (for example, *C:\flexlm*), or use the Browse button to select the directory.
7. In the Path To The License File box, enter the path of the directory where the license file is located (for example, *C:\flexlm*), or use the Browse button to select the directory.
8. In the Path To The Debug Log File box, enter the path of the directory where you want FLEXlm to create the debug file, or use the Browse button to select a directory.
9. Check the User Services check box. The Start Server At Power Up check box becomes enabled.
10. Check the Start Server At Power Up check box.
11. Click the Save Service button and click Yes at the prompt.
12. Select the Start/Stop/Reread tab.
13. Click Start Server.

2.3.10. Troubleshooting the FLEXlm License Manager

For general troubleshooting information, refer to the FLEXlm FAQ at <http://www.macrovision.com/products/flexlm/index.shtml>. It describes common problems, including the majority that have been reported to MÄK support engineers. Please consult the FAQ first when you have a problem.

Unable to Get a License

If you are unable to get a license for a MÄK application and the FLEXlm log file (*LOG*) indicates that the port is in use, it is possible that you have more than one lmgrd or maklmgrd processes running. It is also possible that an application that was using a license is thought to have terminated, but is still alive. To verify and resolve these possibilities:

1. See if more than one lmgrd or maklmgrd process is running.

On Windows, check the Task Manager. (To open the Task Manager, press Ctrl-Alt-Delete and click Task Manager.)

On UNIX, enter the following commands:

```
ps -aux | grep lmgrd
ps -aux | grep maklmgrd
```

If more than one instance of either process is running, write down the process ids (PID).



Some UNIX systems use `ps -aux`, others `ps -ef` to check processes. Use the command appropriate for your operating system.

2. If there are old processes present that may be using a license, kill them.

On UNIX, kill a process with the kill command, for example:

```
kill -9 1385 878
```

where 1385 and 878 are process IDs.

On Windows, select a process in the Task Manager, Processes tab and click End Process.

It is important that you kill all the processes to ensure a clean restart for the license server.

3. See if more than one lmgrd or maklmgrd process is running. Check the Task Manager. (To open the Task Manager, press Ctrl-Alt-Delete and click Task Manager.)
4. If there are old processes present that may be using a license, kill them. Select a process in the Task Manager, Processes tab and click End Process. It is important that you kill all the processes to ensure a clean restart for the license server.

5. If in step 2 you killed the license server, start it with the `runLm` command.

Preventing Multiple License Manager Processes

We recommend that you keep the license server running and that you not start and stop it as you run applications. If you prefer to stop the license server when you are not using it, always use `lmdown` to stop it.

Whenever you start the license manager, first run `lmstat` to be sure that there is not already a license server process running. Following this practice will ensure that you do not start multiple license servers.

License Manager Cannot Find MÄK License Management Executable

If your license manager gives you an error message indicating that it cannot find the license management executable, try putting the pathname in the license file as follows:

1. Open the license file (*mak.lic*) in a text editor.
2. Edit the license file to specify the location of the *maklmgrd* executable on the server machine. The file supplied by MÄK represents this location with a dot as shown below:

```
DAEMON maklmgrd .
```

Replace the dot with your absolute path to *maklmgrd*, for example:

```
DAEMON maklmgrd "C:\flexlm\maklmgrd"
```



On PCs, you must enclose the path in quotes.
The drive letter must be uppercase.

The License Manager Reports an Unsupported Product

If you receive an error message indicating an unsupported product, you are trying to use a MÄK product for which you do not have a license.

If you are trying to run your application with an RTI other than the MÄK RTI and you get the unsupported product message, it is possible that your application is finding the DLLs for an unlicensed version of the MÄK RTI before it finds the other RTI. To fix this problem do the following:

- ♦ Make sure that the other RTI path precedes the MÄK RTI path in your path environment variable.
- ♦ Check the directory in which the application is located to be sure that there are no DLLs from the MÄK RTI in that directory.

2.4. Installing an RTI

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

i

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 MÄK products, see the release notes for your MÄK application or the Product Versions page on the MÄK Technologies web site at: http://www.mak.com/product_version.php.

2.4.1. Installing the MÄK RTI

To install the MÄK RTI, follow the instructions in Chapter 2 of *MÄK RTI Reference Manual*. (If you do not have a printed version of the manual, you can view the PDF versions in *makrtix.x/doc*.)

Configuring Your System to Use the MÄK RTI

The RTI dynamic libraries need to 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:

- ♦ For IRIX6n32 use LD_LIBRARYN32_PATH
- ♦ On other UNIX platforms use LD_LIBRARY_PATH
- ♦ On Windows, use PATH.

The MÄK 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 MÄK RTI

In many cases, you do not need an RTI server, such as the rtiexec, or a central application to use the MÄK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MÄK RTI.) Be sure the FLEXlm license server is running, then start the applications, and they should be able to communicate. (For information about the license server, see “Setting Up and Using the FLEXlm License Manager”.) See *MÄK RTI Reference Manual* for information about the rtiexec and changing the networking parameters such as multicast addresses and UDP ports.

2.4.2. Installing the DMSO RTI

DMSO does not distribute the DMSO RTI any more, but you may be able to obtain it from other sources.

Consult the DMSO RTI Release Notes to determine version information and whether or not any patches are required. Install the RTI according to the instructions provided.

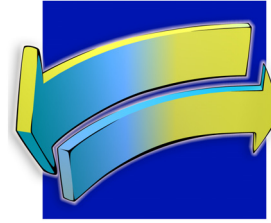
Configuring Your System to Use the DMSO RTI

Follow the configuration instructions provided by your version of the DMSO RTI.

Running Applications Using the DMSO RTI

To use the DMSO RTI, you need to run the RTI executive (rtiexec). This executable is in the *bin* directory or a subdirectory. Run the rtiexec only on one machine, regardless of how many HLA applications you are running.

Once the rtiexec is running, you can run HLA applications.



3

Using the Gateway

This chapter explains how to configure the Gateway and use it.

Overview	3-3
Setting Up the RTI Environment.....	3-3
Starting the Gateway.....	3-3
Starting the Gateway in UNIX.....	3-3
Starting the Gateway on a PC.....	3-4
Starting Message Translation Immediately After Startup.....	3-5
Gateway Startup Options	3-5
Closing the Gateway	3-6
Configuring the Gateway.....	3-6
Configuring the Gateway in the GUI.....	3-7
Specifying the FOM Mapper and FED File to Use.....	3-8
Picking the Ethernet Card to Use	3-9
Translating DIS and HLA Messages.....	3-10
Viewing Objects and Interactions.....	3-10
Enabling and Disabling the Display of Object Data	3-11
Absolute Timestamp Mode	3-12
Forcing Updates	3-12
Stopping Message Translation.....	3-12
Choosing the Objects and Interactions to Translate	3-13
Filtering the Display by Object or Interaction Type.....	3-13
Disabling Translators by Default	3-14
Filtering Translation of Individual Objects	3-14
Translating Incomplete Messages	3-16
Parsing Unknown Global IDs.....	3-16
Specifying the Number of Received Interactions to Display.....	3-16

Using the Gateway

Debugging Translations	3-17
Turning on Debugging by Default	3-17
Dumping Text to a File	3-18
Auto Restart.....	3-18

3.1. Overview

The general steps for using the Gateway are as follows:

1. Set up your RTI environment.
2. Start the Gateway application.
3. Configure the DIS and HLA connections. (You can also configure the Gateway with a configuration file or with command line options.)
4. Optionally, edit the default set of objects and interactions that get translated.
5. Start translating messages.

3.2. Setting Up the RTI Environment



MÄK Gateway does not include an RTI. You must purchase and install a supported RTI to use the Gateway.

To use the Gateway you must set up an RTI environment. See [“Installing an RTI,”](#) on page 2-15 for information about setting up your environment with either the MÄK RTI or DMSO RTI.

- If you are using the DMSO RTI-NG or are using MÄK RTI features that require it, start the *rtiexec*.

3.3. Starting the Gateway

This section explains how to start the Gateway on the different operating systems it supports. The Gateway supports the HLA 1.3 specification and HLA 1516 specification. You start different executables based on the specification being used in the federation that you are translating.

3.3.1. Starting the Gateway in UNIX

To start the Gateway:

1. In a command shell, change directory to the *gateway/bin13* or *gateway/bin1516* directory, depending on which HLA specification you are using.
2. Enter:

```
Gateway [startup_options]
```

The Gateway window opens ([Figure 3-1](#)). The Gateway startup options are described in [“Gateway Startup Options,”](#) on page 3-5.

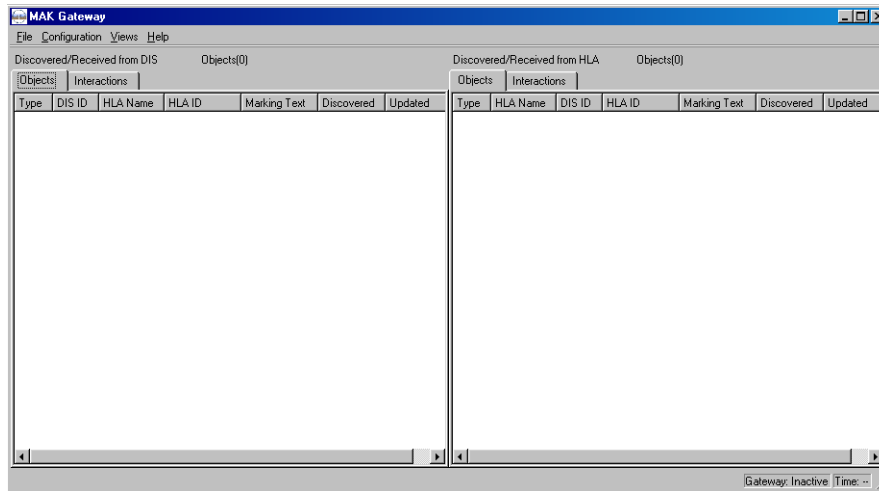


Figure 3-1. Gateway window

Running without a GUI

On Linux, you can run the Gateway without the GUI. If you run the Gateway without the GUI, the Gateway receives and send updates at the interval specified by `pollInterval`. The default interval is 0.001 seconds. Translations start automatically.

- To run the Gateway without the GUI, set the `showGUI` parameter to 0.
- To exit the Gateway when it runs in the background, enter `q` in the command window.

3.3.2. Starting the Gateway on a PC

- To start the Gateway, on the Start menu, choose Programs → MÄK Technologies → Gateway *X.X* HLA 13 or Gateway *X.X* HLA 1516, depending on which HLA specification you are using.

The Gateway window opens ([Figure 3-1](#)).

If you want to start the Gateway with startup options, start it from a command prompt or create a shortcut icon for it on your desktop.

3.3.3. Starting Message Translation Immediately After Startup

At the beginning of this chapter, we list several steps that you may need to take between the time you start the Gateway and the time you start translating state update messages. However, if you have specified all the configuration options that you want in the *gateway.mtl* file or with startup options, you can specify that the Gateway start translating messages as soon as it opens.

- To start translating messages at startup, start the Gateway with the **-S** startup option.

3.3.4. Gateway Startup Options

The Gateway startup options are as follows:

```
Gateway [-A IP_address
        -a application_number
        -b send_port
        -C
        -e exercise_ID
        -F federate_name
        -f FED_file
        -H heartbeat
        -h
        -n notify_level
        -P port
        -R RPRFOM_version
        -S
        -s site_ID
        -T polling_interval]
        -x federation_execution_name
```

where:

- A *IP_address*** specifies the DIS destination address.
- a *application_number*** specifies the DIS application number.
- b *send_port*** specifies the Gateway send port.
- C** opens a console window (Windows only.)
- e *exercise_ID*** specifies the DIS exercise ID.
- F *federate_name*** specifies the federate name that the Gateway is using.
- f *FED_file*** specifies the FED file for the exercise.
- H *heartbeat*** specifies the heartbeat interval, in seconds.
- h** displays the list of startup options.
- n *notify_level*** specifies the notification level. The range for *notify_level* is 0-3, with 0 providing the least amount of information.

- P *port* specifies the DIS port.
- R *RPRFOM_version* specifies the RPR FOM version being used.
- S specifies that the Gateway starts translating messages as soon as it starts.
- s *site_ID* specifies the DIS site ID.
- T *polling_interval* specifies the frequency, in seconds, with which the Gateway checks the exercise connection.
- x *federation_execution_name* specifies the HLA federation execution.

3.3.5. Closing the Gateway

- To close the Gateway, choose File → Quit.

3.4. Configuring the Gateway

Before you start translating update messages, you must configure the Gateway to work with the DIS exercise and HLA federation execution that you are participating in.

You can configure the Gateway using startup options, the *gateway.mtl* file, or through the Gateway graphical user interface. Startup options override the settings in the configuration file. Changes made in the GUI override startup options. Appendix A, [Gateway MTL Parameters](#) describes the parameters in *gateway.mtl*. [Table 3-1](#) maps the equivalent MTL parameters, startup options, and GUI configuration options.

Table 3-1: Configuration option mappings

MTL Parameter	Startup Option	GUI Option
disPort	-P	Port
disExerciseld	-e	Exercise ID
disSiteld	-s	Site ID
disApplicationNum	-a	Application Number
disDestAddr	-A	Dest IP Address
hlaFedName	-F	Federate Name
hlaExecName	-x	Exercise Name
hlaFedFileName	-f	FED File Name
hlaRPRFOMVersion	-R	RPR FOM Version
gatewaySendPort	-b	
heartbeatInterval	-H	
notifyLevel	-n	
pollInterval	-T	
autoStart	-S	

3.4.1. Configuring the Gateway in the GUI

To configure the Gateway:

1. Choose Configuration → Preferences. The Preferences dialog box (Figure 3-2) opens.

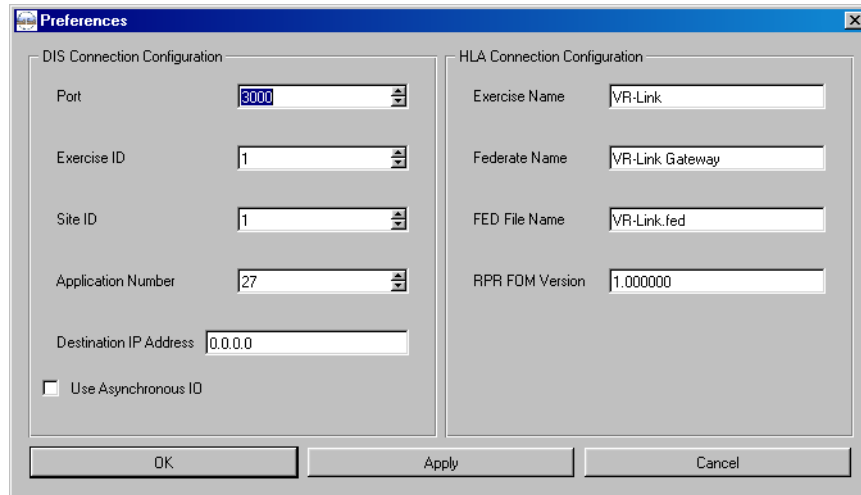


Figure 3-2. Preferences dialog box

2. Specify the DIS and HLA settings for your exercise.



You cannot apply changes made in the Preferences dialog box while the Gateway is translating update messages.

The following table describes the configuration options in the Preferences dialog box.

Table 3-2: Preferences dialog box parameters (Sheet 1 of 2)

Configuration Option	Description
Port	Specifies the port to send/receive network PDUs on. Default: 3000.
Exercise ID	The DIS exercise ID.
Site ID	The DIS site ID.
Application Number	The DIS application number. It is used for creating entity IDs for PDUs originating from the Gateway. It must be unique.
Dest IP Address	Specifies the address of outgoing DIS PDUs. If set to 0.0.0.0, the Default broadcast address is used.
Federate Name	The name of this federate. Default: VR-Link Gateway.

Table 3-2: Preferences dialog box parameters (Sheet 2 of 2)

Configuration Option	Description
Exercise Name	The name of the federation execution. Default: VR-Link.
FED File Name	The name of the FED file being used for the HLA federation execution. It must be the complete name including the three char extension (.fed).
RPR FOM Version	The version of the RPR FOM that you are using. Default: 1.0.

3.4.2. Specifying the FOM Mapper and FED File to Use

This section explains how to specify the FOM Mapper to use with the Gateway. You must specify a FED file that matches the FOM Mapper.

As noted elsewhere in this manual, the Gateway has built-in support for the RPR FOM and VR-Link's ability to support alternative FOMs through FOM Mappers. The Gateway supports different versions of the RPR FOM using FOM Mappers that come with the Gateway and associated FED files. The provided FOM Mappers and FED files are as follows:

- ♦ RPR FOM 0.5 – *VR-Link05.fed*
- ♦ RPR FOM 0.7 – *VR-Link07.fed*
- ♦ RPR FOM 0.8 – *VR-Link08.fed*
- ♦ RPR FOM 1.0 – *VR-Link10.fed* and *VR-Link.fed*
- ♦ RPR FOM 2.0 draft 6 – *VR-Link20006.fed*
- ♦ RPR FOM 2.0 draft 14 – *VR-Link20014.fed*
- ♦ RPR FOM 2.0 draft 17 – *VR-Link20017.fed*

The default FOM used is RPR FOM 1.0. To use one of the other versions of the RPR FOM, specify the FOM version and FED file in the Preferences dialog box, with the `-f` and `-R` startup options, or with the `hlaFedFileName` and `hlaRPRFOMVersion` parameters in *gateway.mtl*.

If you want to use a FOM Mapper that you have written, you must specify the shared library (DSO or DLL) that contains your FOM Mapper and, optionally, any initialization data the library requires. Specify the FOM Mapper library with the `hlaFomMapperLibName` parameter in *gateway.mtl*. Specify the initialization string with the `hlaFomMapperLibInitData` parameter.

For example, if you write a FOM Mapper contained in a shared library called *myFomMapLib.dll*, which requires the string "ABC" as an initialization parameter, specify the following entries in the *gateway.mtl* file.

```
hlaFomMapperLibName "myFomMapLib.dll"
hlaFomMapperLibInitData "ABC"
```

3.4.3. Picking the Ethernet Card to Use

It is common to use the Gateway on a machine with two NIC cards. Typically this configuration is chosen when there is one network for HLA traffic and one network for DIS traffic. Due to the nature of HLA, you must consult your RTI documentation for specifying the HLA network. The DIS network can be configured using *gateway.mtl*. By default the Gateway picks the first network interface it finds for outgoing and incoming DIS traffic. If you want to use a different network, do the following:

- ♦ In *gateway.mtl* set **disDestAddr** to the broadcast address for the DIS Network. For example, if you have a class C network with a subnet mask of 255.255.255.0, and your DIS network card is 158.123.43.54, then set **disDestAddr** to 158.123.43.255.



This is only an example. If you do not know your broadcast address, consult your network administrator.

-
- ♦ In *gateway.mtl* set **disMcasDevice** to the IP address for your DIS network card. In the previous example that would be 158.123.43.54. The **disMcasDevice** parameter is used for both multicast traffic and for configuring your network card. If you are unsure of your IP address, consult your network administrator.

3.5. Translating DIS and HLA Messages

After you configure the Gateway for your exercise, you need to tell it to start translating DIS and HLA messages.

- To start translating messages, choose File → Start Translations.

The Gateway status indicator changes from Inactive to Active (Figure 3-3). The timer displays the Gateway simulation time, in seconds. This time is included in all messages that the Gateway generates.

3.5.1. Viewing Objects and Interactions

The Gateway displays lists of the objects and interactions that it discovers (Figure 3-3). Interactions and objects are listed on separate tabs and are also separated by simulation standard (DIS and HLA). As objects join and leave the exercise, the Gateway updates the display so that it only shows currently simulated objects. The Gateway does not automatically clear the Interactions list. As more interactions are discovered, they are added to the list.

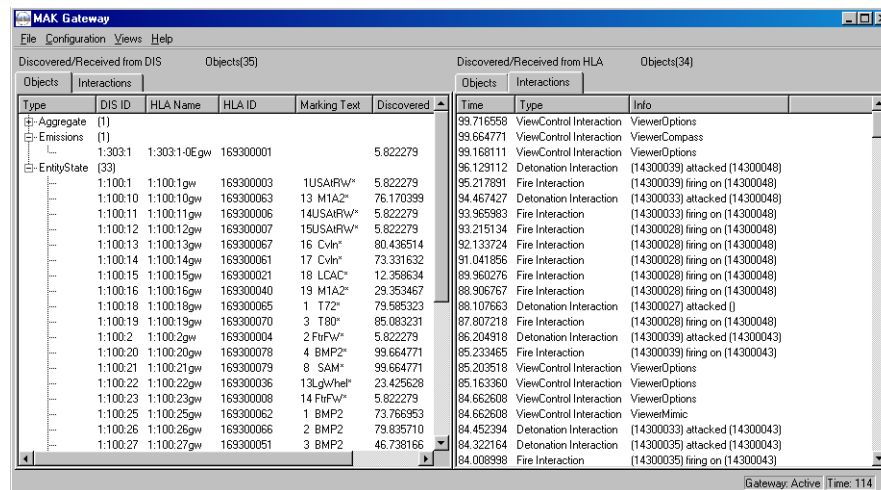


Figure 3-3. Discovered objects and interactions

- To view objects and interactions, select the Interactions or Objects tab for the protocol whose messages you want to view.

Handling Incomplete HLA Objects

The Gateway does not send DIS PDUs for an object until it has all of the information it needs to complete the PDU. Since HLA object updates do not contain the complete description of the object's state, it is possible that the Gateway will discover an object, but not have enough information to translate its state to DIS. In such a case, the object display shows *inc* to the left of the entry in the list. When the PDU for an object is sent, the notation is removed.

Handling Unknown Objects

When the gateway discovers an object that it cannot translate, and which it will drop, it lists the object in the GUI with "dr" in the left-most column.

The reasons for which an object may be dropped include:

- ♦ An inability to map a DIS object to a FOM object (because the FED file does not support it)
- ♦ A name conflict.

Sorting the Object List by Column

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

Clearing the Interactions List

If you want to, you can clear the Interactions list.

- To clear the Interactions list, choose Views → Interactions → Clear Interactions.

3.5.2. Enabling and Disabling the Display of Object Data

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

- ♦ Objects – show or hide all objects
 - ♦ HLA ID – show or hide the HLA ID column
 - ♦ Updates – show or hide the Updated column
 - ♦ Marking text – show or hide the Marking Text column.
- To enable or disable display of object data, choose Views → Objects → *option*, where *option* is one of the following:
 - Show Objects
 - Show Updates
 - Show HLA IDs
 - Show Marking Text.

or right-click on white space in an object list window and choose one of the options from the menu.

3.5.3. Absolute Timestamp Mode

In absolute timestamp mode, the Gateway sends PDUs or HLA interactions with absolute timestamps. The timestamps are generated based the system clock of the computer on which the Gateway is running. Therefore, the system clock of the computer running the Gateway must be synchronized with the other computers in the exercise. You enable or disable absolute timestamps with the `timeStampAbsolute` parameter in the `gateway.mtl` file, as follows:

```
(setqb timeStampAbsolute mode)
```

where *mode* is:

- ♦ 0 = Relative timestamps
- ♦ 1 = Absolute timestamps.

Default: 0.

3.5.4. Forcing Updates

You can force the Gateway to send complete object updates for DIS or HLA. When you force updates, they are sent regardless of which fields have been updated, and whether or not they have exceeded their dead reckoned thresholds.

- To force the Gateway to send complete updates to HLA, set `forceHlaUpdates` to 1.
To force the Gateway to send complete updates to DIS, set `forceDisUpdates` 1.

3.5.5. Stopping Message Translation

You can stop the Gateway from processing state update messages (PDUs).

- To stop the processing of update messages, choose File → Stop Translations.

3.6. Choosing the Objects and Interactions to Translate

By default, the Gateway translates all objects and interactions. If you do not want to process all incoming messages, you can dynamically filter translations by object or interaction type or by individual object.

3.6.1. Filtering the Display by Object or Interaction Type

The Translation Manager lets you filter translations by object or interaction type.

To specify the objects and interactions that the Gateway translates:

1. Choose Configuration → Translations. The Translation Manager opens ([Figure 3-4](#)). The interactions and objects that you can translate are listed in a tree-style hierarchy.

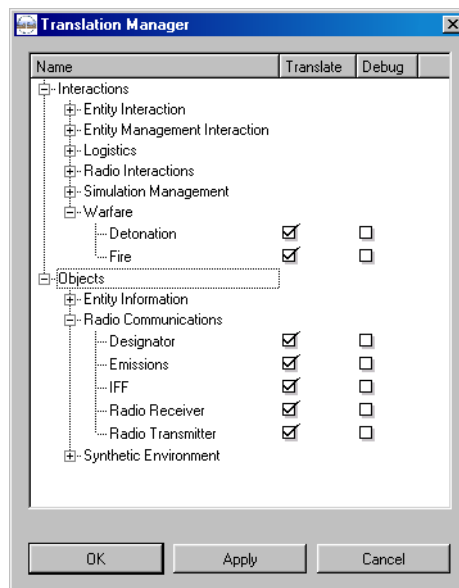


Figure 3-4. Translation Manager

2. Expand the hierarchy to view the interactions and objects available for translation.
3. Check the box in the Translate column for each object or interaction that you want to translate.
4. Check the box in the Debug column if you want to view the contents of the messages that the Gateway is translating.
5. Click OK.



You can change the list of objects and interactions that the Gateway translates while the Gateway is translating messages. Changes take place immediately.

3.6.2. Disabling Translators by Default

The `disabledTranslators` parameter lets you specify translators to be off by default, for example:

```
(setq disabledTranslators
  (list
    "Fire"
    "Detonation"
    "EntityState"
    "Acknowledge"
  )
)
```

You can still change settings for all translators in the GUI.

3.6.3. Filtering Translation of Individual Objects

You can filter out individual objects from being translated. If an object is being filtered, it is marked with an **f** to the left of its entry in the list ([Figure 3-5](#)).

[-] Emissions	(1)				
[-] EntityState	(28)				
	1:100:1	1:100:1gw	39300001	1USARW*	2.403E
	1:100:10	1:100:10gw	39300012	13 M1A2*	3.383C
	1:100:11	1:100:11gw	39300004	14USARW*	2.403E
	1:100:12	1:100:12gw	39300005	15USARW*	2.403E
f	1:100:13			16 Cvin*	6.383E
f	1:100:14			17 Cvin*	6.383E
f	1:100:15			18 LCAC*	3.383C
f	1:100:16			19 M1A2*	3.383C
f	1:100:18			1 T72*	3.383C
f	1:100:19			3 T80*	3.383C
f	1:100:2			2 FtrFW*	2.403E
f	1:100:20			4 BMP2*	3.383C
f	1:100:21			8 SAM*	3.383C
	1:100:22	1:100:22gw	39300034	13LgW/hel*	20.237
	1:100:23	1:100:23gw	39300006	14 FtrFW*	2.403E
	1:100:25	1:100:25gw	39300019	1 BMP2	3.383C
	1:100:26	1:100:26gw	39300020	2 BMP2	3.383C
	1:100:27	1:100:27gw	39300021	3 BMP2	3.383C
	1:100:28	1:100:28gw	39300022	16 T80*	3.383C
	1:100:3	1:100:3gw	39300008	3 M1A2*	3.383C
	1:100:35	1:100:35gw	39300038	20TowMsl*	31.39C
	1:100:4	1:100:4gw	39300009	4 M1A2*	3.383C

Figure 3-5. Object list with filtered objects

Notes

- If you filter an object that has an embedded system, such as an emitter, the embedded object also gets filtered. Similarly, if you filter an embedded system, its parent object gets filtered.
- Filtering objects does not affect translation of interactions. If an interaction is sent that would ordinarily affect an object that is being filtered out, the interaction has no effect on the object.

To filter or unfilter translation of objects:

1. In the object list, select the objects you want to filter or unfilter.
2. On the right-click menu, choose Filter Selected to stop translating the objects. Choose Unfilter Selected to resume translating the objects.

i

You can also add or remove objects in the Object Filters dialog box. See the following section, [Managing Filtered Objects](#).

Managing Filtered Objects

You can manage the objects being filtered. They are listed in the Object Filters dialog box by:

- ♦ DIS object – any DIS object being filtered
- ♦ HLA RTI Object ID – HLA objects listed by the object ID assigned by the RTI
- ♦ HLA RPR FOM Entity ID – for federations using the HLA RPR FOM, objects listed by the Entity ID attribute.

You can remove listed objects from the list of filtered objects and you can add individual objects to any of the lists.

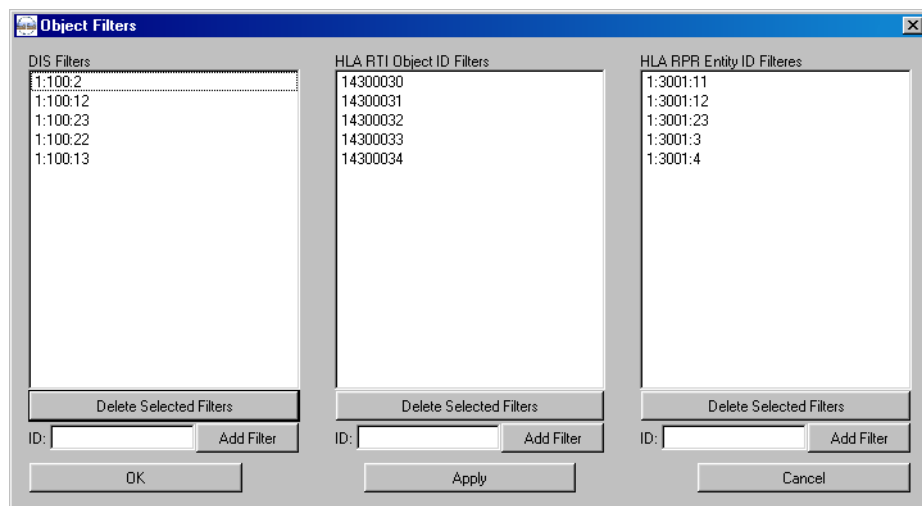


Figure 3-6. Object Filters dialog box

To delete a filtered object from a list:

1. Choose Configuration → Object Filters. The Object Filters dialog box opens.
2. Select the object in the appropriate list. (You can select multiple objects too.)
3. Click Delete Selected Filters.

You can add objects to the lists of filtered objects.

- To add an object to one of the filtered objects lists, enter the object's object ID in the ID box and click Add Filter.

3.6.4. Translating Incomplete Messages

You can choose to translate messages that would normally be marked incomplete. If you set `strictDiscoveryCriteria` to 0 (zero), the Gateway translates objects that would otherwise be marked as incomplete. For example, entities at the location 0,0,0 would normally be marked incomplete. If `strictDiscoveryCriteria` is set to 0, such an entity would get translated.

3.6.5. Parsing Unknown Global IDs

Some objects and interactions reference another object. For example, an emitter system references the entity to which it is attached. Typically, if the global object ID is unknown (because the entity to which the emitter system is attached has not been published), the Gateway translates the ID as either 0:0:0 (for DIS) or an empty string (for HLA). You can configure the Gateway so that it attempts to parse the unknown ID to construct a meaningful ID in the other protocol. If an unknown ID is passed through the Gateway from DIS, the Gateway constructs an HLA name by appending gw to the DIS ID. For example, the ID 3:4:5 becomes 3:4:5gw. If an unknown ID is passed through the Gateway from HLA, the Gateway can only parse it if it begins with x:y:z, where x, y, and z are integers. If it is parseable, the Gateway constructs a DIS ID by converting the x:y:z string to the DIS ID, x:y:z. For example, the HLA ID 1:2:3-gw becomes DIS ID 1:2:3. If the string cannot be parsed, the ID is translated as 0:0:0.

- To enable parsing of unknown global IDs, set `parseUnknownGlobalIds` to 1 in *gateway.mtl*.

3.6.6. Specifying the Number of Received Interactions to Display

As the list of received interactions grows, the Gateway's memory usage increases. By default, the Gateway displays the most recent 15 interactions received. You can specify the number of interactions to display.

- To specify the number of interactions to display, edit the `numberOfInteractionsShown` parameter. To display all interactions received, set `numberOfInteractionsShown` to -1.

3.7. Debugging Translations

If you check the Debug box for an interaction or object in the Translation Manager, the Gateway prints information to the *gateway.log* file and the console (if enabled.) The information printed is determined by the **notifyLevel** setting in *gateway.mtl*. To see message information, set **notifyLevel** to at least 2. Messages such as the following are printed:

```
Gateway Packet: ----- ORIG-(Detonation) DIS at 16.999478
From:          1:15:1
Target:        0:0:0
Munition:      1:15:10
Event:         1:15:2
WorldLocation: { -2661388.272, -4359496.289, 3807301.109}
Velocity:      { 0.000, 0.000, 0.000}
Result:        DetResGroundImpact (3)
FuseType:      FuzeOther (0)
MunitionType:  2:2:225:1:3:0:0
Quantity:      1
Rate:          0
WarheadType:   WarheadOther (0)
EntityLocation: { 0.000, 0.000, 0.000}
NumArtParams:  0
Gateway Packet: ===== NEW-(Detonation) HLA =====
From:          1:15:1gw
Target:        1:15:10gw
Munition:      1:15:10gw
Event:         2:Gateway
WorldLocation: { -2661388.272, -4359496.289, 3807301.109}
Velocity:      { 0.000, 0.000, 0.000}
Result:        DetResGroundImpact (3)
FuseType:      FuzeOther (0)
MunitionType:  2:2:225:1:3:0:0
Quantity:      1
Rate:          0
WarheadType:   WarheadOther (0)
EntityLocation: { 0.000, 0.000, 0.000}
```

3.7.1. Turning on Debugging by Default

All translators have debug output turned off by default. You can now use the **debuggedTranslators** parameter to specify translators that you want to start with debugging turned on, for example:

```
(setq debuggedTranslators
  (list
    "Detonation"
    "Designator"
    "IFF"
    "LoggerControl"
  )
)
```

If the translator is not active during translation, there will not be any output.

3.7.2. Dumping Text to a File

When an interaction, or an object update is received by the gateway, the GUI is updated to reflect the update. However, for debugging, its often useful to dump this information to the Gateway log file. The `dumpGuiToText` parameter allows you to do this.



Dumping text to the log file substantially increases the amount of file writing that the Gateway does and may negatively affect performance.

```
(setqb dumpGuiToText mode)
```

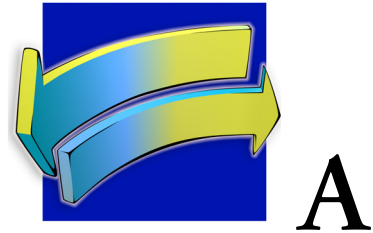
where *mode* is one of the following:

- ♦ 0 = Do not dump text
- ♦ 1 = Dump text.

3.8. Auto Restart

If the Gateway loses its connection to the RTI, it stops translating messages. You can start translating again manually when the Gateway reconnects to the RTI or you can specify that the Gateway try to restart translations automatically.

- To specify that the Gateway try to restart automatically, set the `autoRestart` parameter is set to 1. The Gateway tries to restart translations every 10 seconds after the translations are stopped.



Gateway MTL Parameters

This chapter describes the parameters in the *gateway.mtl* file.

MÄK Technologies Lisp (MTL) Files.....	A-2
Using Environment Variables in an MTL File	A-2
Specifying Lists of Lists.....	A-3
Using Conditional Statements	A-3
Gateway Configuration Parameters	A-4

A.1. MÄK Technologies Lisp (MTL) Files

Many of the configuration files for MÄK Technologies products are ASCII text files that use MÄK Technologies LISP (MTL) to encode configuration information. MTL files have the extension *.mtl*. You can edit them in any text editor. Be sure that your text editor uses the proper end-of-line characters for your platform.

Most parameter entries take the format:

```
(setq parameter_name value)
```

For example:

```
(setq EnableSound 1)
```

Or:

```
(setqb parameter_name value)
```

The `setqb` syntax indicates a bound variable.

To comment out a line, precede the text with two semi-colons (`::`). You can also add a comment at the end of a line by preceding it with two semi-colons.

In most cases, the parameter names are known to the MÄK application and the MTL commands simply assign values to them. However, you can use the `setq` command to create arbitrary symbolic names or aliases that are then used in later commands. For example, in the following two commands (from the MÄK Stealth), the symbolic name `VaporTrail` is assigned to an `OpenFlight` file. Then the symbolic name is used in the `trail-map` command. The name `VaporTrail` has no intrinsic meaning to the Stealth. It is given meaning by the `setq` command, which allows it to be used later in the `trail-map` command.

```
(setq VaporTrail (list "../data/models/makTrails/Vapor-Trail" "VaporTrail.flt" 6.0 1.5 1.5 4.0 1.5 0.5))
(trail-map (list 1 2 -1 0 -1 -1 -1)(list VaporTrail 0.0 0.0 0.0))
```

In addition to the `setq` command, a commonly used command is the `load` command. This command instructs the application to load the file specified, for example:

```
(load mtl-path "params.mtl")
```

A.1.1. Using Environment Variables in an MTL File

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

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

For example:

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

A.1.2. Specifying Lists of Lists

The example of setting a symbolic name in a previous section uses a list. You can also specify a list of lists, for example:

```
(trail-map (list 1 2 -1 -1 -1 -1 -1)
  (list vaporTrail -8.0 -3.0 0.0)
  (list vaporTrail -8.0 3.0 0.0)
)
```

A parameter entry can also have multiple individual lists:

```
(trail-map (list 1 2 -1 0 -1 -1 -1) (list VaporTrail 0.0 0.0
  0.0))
```

A.1.3. Using Conditional Statements

MTL commands can be conditional. Conditional statements are used frequently when parameters are different for DIS and HLA. A simple conditional statement has the format:

```
(if (condition) (statement_to_evaluate))
```

If you have more than one statement to evaluate, use a sequence block:

```
;; HLA specific parameters
(if (equal HLA 1)
  (sequence
    (setqb fomMapperLib "")
    (setqb fomMapperInitData "")
    (setqb rprFomVersion 1.0)
    (setqb pathToSublistFile "sublist.mtl")
    (setqb ignoreAdvisories 0)
    (setqb fedLookahead 1.0)
  )
)
```

A.2. Gateway Configuration Parameters

When you start the Gateway, it automatically loads the *gateway.mtl* configuration file. [Table A-1](#) describes the configuration parameters in *gateway.mtl*.

Table A-1: Gateway configuration parameters (Sheet 1 of 7)

Parameter	Description
autoStart	Tells the Gateway to automatically start translating on startup. ♦ 0 = Do not start translating automatically. ♦ 1 = Start translating automatically. Default: 0.
disApplicationNum	The DIS application number. It is used for creating entity IDs for PDUs originating from the Gateway. It must be unique.
disDestAddr	Specifies the address of outgoing DIS PDUs. If set to 0.0.0.0, the Default broadcast address is used.
disExerciseId	The DIS exercise ID.
disPort	Specifies the port to send or receive network PDUs on. Default: 3000.
disSiteId	The DIS site ID.
heartbeatInterval	Specifies the number of seconds between heartbeats of published entities. Default: 5.0.
hlaExecName	The name of the federation execution. Default: VR-Link.
hlaFedFileName	The name of the FED file being used for the HLA federation execution. It must be the complete name including the three char extension (<i>.fed</i>).
hlaFedName	The name of this federate. Default: VR-Link Gateway.
hlaRPRFOMVersion	The version of the RPR FOM that you are using. Default: 1.0.
logFileName	Specifies the name of the file to dump Gateway output to. This flag is only valid on Windows. Default: <i>gateway.log</i>
pollInterval	Specifies the interval, in seconds, between each call to drainInput() on the exercise connection.
timeoutInterval	Specifies the number of seconds before a reflected entity times out and the Gateway removes it. Default: 10.0.

Table A-1: Gateway configuration parameters (Sheet 2 of 7)

Parameter	Description
timeStampAbsolute	Specifies use of absolute timestamps on PDUs or interactions. ♦ 0 = Do not use absolute timestamps. ♦ 1 = Use absolute timestamps. Default: 0.
useConsoleOutput	Specifies creation of a Windows console for all Gateway output. This flag is only valid on Windows. ♦ 0 = Do not create a console. ♦ 1 = Create a console. Default: 0.
Debug Options	
debugEntityIdMapping	Specifies whether or not to map HLA entity IDs to DIS entity IDs in the debug output. ♦ 0 = Do not map HLA entity IDs to DIS entity IDs. ♦ 1 = Map HLA entity IDs to DIS entity IDs. Default: 0.
debugGlobalIdMapping	Specifies whether or not to map HLA global object designators to DIS entity IDs in the debug output. ♦ 0 = Do not map HLA global object designators to DIS entity IDs. ♦ 1 = Map HLA global object designators to DIS entity IDs. Default: 0.
dumpGuiToText	Specifies whether or not to write updates to a log file. This information is already displayed in the GUI, but using <code>dumpGuiToText</code> makes a better record for debugging.  The notify level must be set to at least 1. ♦ 0 = Do not dump text. ♦ 1 = Dump the text. Default: 0
notifyLevel	Specifies the level of notification messages sent by VR-Link. Range: 0-3. Default: 1.

Table A-1: Gateway configuration parameters (Sheet 3 of 7)

Parameter	Description
Advanced Configuration	
debuggedTranslators	Specifies a list of translators for which debugging is enabled when the Gateway starts up. You can still change all debug settings in the GUI.
disabledTranslators	Specifies a list of translators to be set to “inactive” when the Gateway starts up. You can still activate these translators in the GUI.
disDrainTimeout	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 disNumPdusToReadPerTimeCheck PDUs at a time, until all PDUs are read or the timeout value is reached, whichever comes first. Default: -1.
disMcastDevice	Specifies the IP address to use for multicast traffic. Default: 0.0.0.0 (the default network address.)
disNumPdusToReadPerTimeCheck	Specifies the number of PDUs to read before checking to see if the amount of time specified by disDrainTimeout has been reached. Default: 100.
disObjectFilters	Specifies a list of DIS Entity IDs for the Gateway to filter out. The Gateway will not publish any object discovered from DIS whose entity ID matches an ID in this list.
disProtocolVersionToSend	The DIS PDU versions to send and receive, where: • 4 = DIS 2.0.4 • 5 = IEEE 1278.1 • 6 = IEEE 1278.1a. Default version to receive: 4, 5, and 6. Default version to send: 5.
disProtocolVersionToRecvMin	
disProtocolVersionToRecvMax	
disSocketReceiveBufferSize	Sets the DIS socket’s buffer received size. If you think the gateway 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 Gateway uses the default size for the operating system.

Table A-1: Gateway configuration parameters (Sheet 4 of 7)

Parameter	Description
disUseAsynchIO	Lets you use asynchronous I/O for the DIS side of the Gateway. When the Gateway uses asynchronous I/O, the PDUs are read from the network in a separate thread. This keeps the Gateway from dropping packets during periods of extremely high load. However, it also adds a small amount of overhead because all packets are put in a queue before being read. Therefore, you should use asynchronous I/O only in an exercise where PDUs arrive in large waves and in which it is critical to not drop PDUs. ♦ 0 = Do not use asynchronous I/O ♦ 1 = Use asynchronous I/O. Default: 0.
forceDisUpdates	When enabled (1), forces the Gateway to send a DIS PDU when it receives an update message for an HLA object. Default: 0.
forceHlaUpdates	When enabled (1), forces the Gateway to send a complete HLA update when it receives a DIS PDU. Default: 0.
gatewaySendPort	Specifies the port the Gateway sends PDUs on. This port is only to be used by the Gateway. It knows that if it receives a message with this port, that it is its own message, so it doesn't translate it. Make sure this port doesn't conflict with others on your network. Default: 2999.
hlaEntityIdObjectFilters	Specifies a list of RPR FOM entity IDs for the Gateway to filter out. The Gateway will not publish DIS objects for any discovered HLA objects whose Entity ID attribute matches an ID in this list.
hlaFomMapperLibInitData	A string that is passed to the FOM mapper library specified in fomMapperLibName. This data is optional. Default: "". See "Specifying the FOM Mapper and FED File to Use," on page 3-8.
hlaFomMapperLibName	The name of a FOM mapper library, if you are using one. Default: "". See "Specifying the FOM Mapper and FED File to Use," on page 3-8.
hlaMaxTickTime	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 hlaMinTickTime, but no more than this value. Default: -1.

Table A-1: Gateway configuration parameters (Sheet 5 of 7)

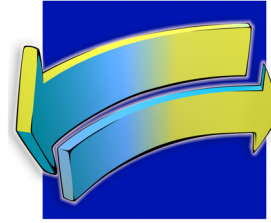
Parameter	Description
hlaMinTickTime	Specifies the minimum amount of time, in seconds, to tick the RTI, before processing incoming messages. Default: 0.0.
hlaObjectFilters	Specifies a list of RTI object ID's for the Gateway to filter out. The Gateway will not publish DIS objects for any discovered HLA objects whose RTI Object ID matches an ID in this list.
hlaUseAdvisories	Specifies whether the Gateway should use RTI advisory messages when determining what attributes to update. ♦ 0 = Send attribute updates regardless of whether or not someone has subscribed to the attribute with the RTI. ♦ 1 = Send attribute updates only if someone has subscribed to the attribute. Default: 1
multicastSubscriptions	Specifies a list of multicast addresses the Gateway should subscribe to for DIS communication.
multicastTimeToLive	Specifies a multicast time to live value. Default: 0 (use the system default)
parseUnknownGlobalIds	Allows identifiers to be sent from HLA to DIS, if the string begins with the site:application:entity triplet, even if the object identified is not known by the DIS simulation.
publisherTickInterval	Specifies the time between ticks of all Gateway publishers, in seconds. In general, you should not change this value.
renameDisObjects	Specifies whether or not to map all DIS entity IDs to match the site/host IDs specified in <i>gateway.mtl</i> . This corrects errors caused by rogue DIS Objects in a DIS->HLA->DIS exercise. ♦ 0 = Do not map IDs. ♦ 1 = Map IDs. Default: 0.
strictDiscoveryCriteria	Enables (0), or disables (1), translation of messages that would otherwise be marked incomplete. Default: 0.
useAstiRadioModes	Enable (1) or disable (0) use of ASTI Radio FOM extensions in which radio host IDs are encoded in the streamTag.

Table A-1: Gateway configuration parameters (Sheet 6 of 7)

Parameter	Description
GUI Options	
autoRestart	Tries to start translation every ten seconds if the connection to the rtiexec is broken. ♦ 0 = Do not try to restart. ♦ 1 = Try to restart. Default: 0.
displayLastUpdateThreshold	Specifies the minimum amount of time between GUI updates for the Updated column. It does not affect the object translation. If you specify 0.0, each object's last update time is displayed in the GUI as soon as the object is updated. For exercises with very large numbers of objects, the performance of the Gateway may be hindered by updating the GUI too frequently. Therefore, updating the display less frequently may improve performance. Default: 1.0.
numberOfInteractionsShown	Specifies the how many or the most recent interactions to show in the display. Reducing the number of interactions shown reduces memory usage. Default: 15.
showGui	Runs the Gateway in the background. Linux only. ♦ 0 = Do not display GUI. ♦ 1 = Display GUI. Default: 1.
showHlalds	Specifies display of the HLA ID column in the Objects list. ♦ 0 = Do not display HLA IDs. ♦ 1 = Display HLA IDs. Default: 1.
showInteractions	Specifies the display of interactions. It does not affect the translation of interactions. ♦ 0 = Do not display interactions. ♦ 1 = Display interactions. Default: 1.
showMarkingText	Specifies display of the Marking Text column in the Objects list. ♦ 0 = Do not display marking text. ♦ 1 = Display marking text. Default: 1.

Table A-1: Gateway configuration parameters (Sheet 7 of 7)

Parameter	Description
showObjects	Specifies display of discovered and published objects. It does not affect the translation of objects. ♦ 0 = Do not display discovered objects. ♦ 1 = Display discovered objects. Default: 1.
showObjectUpdates	Specifies display of the last time that the Gateway received an update for this object. ♦ 0 = Do not display the update time. ♦ 1 = Display the update time. Default: 1.



MÄK Products Glossary

This glossary defines terms used in MÄK 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.

In the Stealth, when eyepoint moves with respect to an entity, we describe it as movement in a body coordinate frame.

channel

A channel is a particular view when you are displaying multiple views of a simulation.

clipping

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

clipping planes

The range of distances from the eyepoint (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.

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.

Defense Modeling and Simulation Office

The executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), which provides a full-time focal point for information concerning Department of Defense modeling and simulation (M&S) activities. Currently the DMSO 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](#).

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

[Defense Modeling and Simulation Office.](#)

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 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, 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 Gateway application connects to the network. State messages are sent through the exercise connection and information about remote entities is received through the exercise connection.

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 also reads 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

Field of view controls the perspective of the eyepoint.

A wide field of view creates an effect like that of a wide-angle camera lens. Objects appear smaller and farther away from the eyepoint, since the eyepoint 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 eyepoint, 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

[Federation Object Model](#).

frame rate

The rate at which the application displays updated images.

ground clamping

A process by which the Stealth 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 through 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. Gateway 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.

heads-up display

A set of indicators and readouts superimposed onto the graphics display. It displays the state of the eyepoint and the locations of other entities. Also called the 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 DOD-wide initiative to provide an architecture to support interoperability and reuse of simulations. The HLA supersedes DIS for the Department of Defense.

HLA

[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 MÄK Logger remotely.

MÄK Technologies Lisp

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

message

A general term used to refer to [DIS PDUs](#) and [HLA interactions](#) and state updates.

MTL

[MÄK 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. 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.

PDU

[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 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.

RTI Initialization Data (RID)

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

[Realtime Platform Reference FOM](#).

RTI

[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.

smooth period

The period of time over which [trajectory smoothing](#) takes place, set to one second by default.

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

Simulation Interoperability Standards Organization

smoothing

A method of ensuring that transitions from an entity's dead-reckoned position to its actual position aren't 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.

terrain following

Causes the eyepoint to maintain a constant distance above the terrain surface. The eyepoint's height changes in tandem with the peaks and valleys as it passes over the geography.

tape

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

timeout

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

timescale

A factor by which time is accelerated or slowed during playback of a 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. When the eyepoint moves with respect to the terrain, we describe it as movement in a topographic coordinate frame.

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. When in UTM mode (the default), the Stealth uses an offset UTM coordinate system, one in which coordinates are expressed with respect to an origin located at the latitude and longitude specified in a configuration file. Typically, this will be the location of the origin of your terrain database.

UTM coordinates

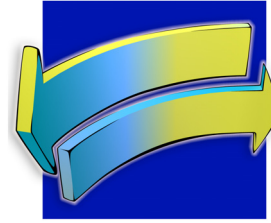
[Universal Transverse Mercator](#) coordinates.

view control messages

A set of programmatic messages that let you control the MÄK Stealth or Logger remotely.

view floor

A minimum height above the terrain in Absolute or Track mode, below which the eyepoint is not allowed to go. The view floor is measured relative to the terrain surface.



Index

Numerics

1516 1-3

A

- A startup option 3-5
- a startup option 3-5
- absolute timestamp 3-12, A-5
- Acknowledge* class 1-9
- Acknowledge PDU 1-9
- Action Request PDU 1-9
- Action Response PDU 1-9
- ActionRequest* class 1-9
- ActionResponse* class 1-9
- Active 3-10
- adding
 - new license file 2-10
 - object to filter list 3-15
- address
 - destination 3-7, A-4
- application number 3-7, A-4
- ApplicationSpecificRadioSignal* interaction 1-10
- asynchronous I/O A-7
- attributes
 - missing 1-4
- auto start 3-5
- automatic restart 3-18
- automatically starting translations A-4
- autoRestart** parameter 3-18, A-9
- autoStart** parameter 3-6, A-4

B

-b startup option 3-5

- background
 - running Gateway in 3-4
- BaseEntity* class 1-7, 1-8
- beam ID 1-11
- BeamID parameter 1-5
- BeamParameterIndex parameter 1-5
- broadcast address 3-7, A-4
- buffer
 - receive size A-6

C

- C startup option 3-5
- class
 - Acknowledge* 1-9
 - ActionRequest* 1-9
 - ActionResponse* 1-9
 - BaseEntity* 1-7, 1-8
 - Collision* 1-8
 - Comment* 1-9
 - CreateEntity* 1-9
 - Data* 1-9
 - DataQuery* 1-9
 - Designator* 1-5, 1-10
 - EmbeddedSystem* 1-10, 1-11
 - EmitterBeam* 1-5
 - EmitterSystem* 1-5, 1-10, 1-11
 - EmittingSystemID* 1-11
 - EventReport* 1-9
 - GroundVehicle* 1-7
 - IFF* 1-5, 1-10
 - IffInterrogator* 1-10, 1-11
 - IffTransponder* 1-10, 1-11
 - JammerBeam* 1-11
 - MunitionEntity* 1-7

- PhysicalEntity* 1-8
- RadarBeam* 1-11
- RadioReceiver* 1-5, 1-10
- RadioTransmitter* 1-5, 1-10
- RemoveEntity* 1-9
- RepairComplete* 1-12
- RepairResponse* 1-12
- ResupplyCancel* 1-12
- ResupplyOffer* 1-12
- ResupplyReceived* 1-12
- ServiceRequest* 1-12
- SetData* 1-9
- StartResume* 1-9
- StopFreeze* 1-9
- clearing interaction list 3-11
- closing
 - Gateway 3-6
- Collision* class 1-8
- Collision PDU 1-7
- column
 - sorting by 3-11
- comment
 - MTL parameter A-2
- Comment* class 1-9
- Comment PDU 1-9
- configuration file 3-17
 - gateway.mtl A-4
 - RTI 2-15
- configuring
 - Gateway 3-6
 - Gateway through GUI 3-7
 - GUI options A-10
 - system for MÄK RTI 2-15
 - time between display updates A-9
- connection
 - losing to RTI 3-18
- console A-5
- conventions
 - manual 1-xi
- Create Entity PDU 1-9
- CreateEntity* class 1-9

D

- data
 - debug 3-17
- Data* class 1-9
- Data PDU 1-9
- Data Query PDU 1-9
- DatabaseIndexRadioSignal* interaction 1-10

- DataQuery* class 1-9
- Debug
 - box 3-17
 - column 3-13
- debug data 3-17
- debugEntityIdMapping* parameter A-5
- debuggedTranslators* parameter 3-17, A-6
- debugging
 - dumping text to a file 3-18
 - translators 3-17
- debugging translations 3-13
- debugGlobalIdMapping* parameter A-5
- deleting
 - object from filtering list 3-15
- Delta State EE PDU 1-11
- Designator* class 1-5, 1-10
- Designator PDU 1-11
- destination address 3-7, A-4
- Detonation PDU 1-8
- DIS
 - filtering objects A-6
 - forcing updates 3-12
 - mapping entity ID to global object
 - designator A-5
 - multicast traffic A-6
 - renaming entity IDs to match site/host A-8
 - using asynchronous I/O A-7
 - version A-6
- DIS object
 - filtering by 3-15
- DIS protocol 1-2
- disabledTranslators* parameter 3-14, A-6
- disabling
 - display of HLA ID A-9
 - display of interactions A-9
 - display of marking text A-9
 - display of objects A-10
 - translators 3-14
 - translators at startup A-6
- disApplicationNum* parameter 3-6, A-4
- discovered object and interaction
 - viewing 3-10
- disDrainTimeout* parameter A-6
- disExerciseld* parameter 3-6, A-4
- disMcastDevice* parameter A-6
- disNumPdusToReadPerTimeCheck* parameter A-6
- disObjectFilters* parameter A-6
- display
 - filtering object data 3-11
- displaying

HLA ID A-9
 interactions 3-16, A-9
 marking text A-9
 objects A-10
displayLastUpdateThreshold parameter A-9
disPort parameter 3-6, A-4
disProtocolVersionToRecvMax parameter A-6
disProtocolVersionToRecvMin parameter A-6
disProtocolVersionToSend parameter A-6
disSitelD parameter 3-6, A-4
disSocketReceiveBufferSize parameter A-6
 Distributed Emission Regeneration protocol
 family 1-11
disUseAsynchIO parameter A-7
 DMSO RTI 2-16
 dropping packets A-6, A-7
dumpGuiToText parameter 3-18, A-5

E

-e startup option 3-5
 EE PDU 1-11
 Electromagnetic Emission PDU 1-11
 embedded system object
 deletion of 1-6
EmbeddedSystem class 1-10, 1-11
 emission 1-11
 emitter beam 1-11
 emitter system 1-11
EmitterBeam class 1-5
 EmitterName parameter 1-5
 EmitterNumber parameter 1-5
EmitterSystem class 1-5, 1-10, 1-11
 emitting system ID 1-11
EmittingSystemID class 1-11
 EmittingSystemID parameter 1-5
 enabling
 debug translators at startup A-6
 display of HLA ID A-9
 display of interactions A-9
 display of marking text A-9
 display of objects A-10
EncodedAudioRadioSignal interaction 1-10
 entity ID 1-8, 1-10
 filtering by 3-15, A-7
 mapping to HLA global object designator A-5
 renaming to match site/host A-8
 Entity Information 1-7
 Entity Management PDU family 1-12
 Entity State PDU 1-7

entity type 1-7
EntityIdentifier parameter 1-5
 environment variable
 MAKLMGRD_LICENSE_FILE 2-9
 RTL_CONFIG 2-15
 Environmental Process PDU 1-12
 EnvironmentProcess class 1-12
 ERP parameter 1-5
 Event Report PDU 1-9
EventReport class 1-9
 executing gateway
 UNIX 3-3
 Windows 3-4
 exercise connection A-4
 exercise ID 3-7, A-4

F

f
 in object display 3-14
 -F startup option 3-5
 -f startup option 3-5
 FDD file 2-15
 FED file 1-3, 2-15, 3-8, A-4
 federate name 3-7, A-4
 federation execution 1-3, 3-8, A-4
 file
 FED 2-15
 gateway.log 3-17
 gateway.mtl 3-6, 3-8, 3-17
 license 2-9
 log 3-18, A-4
 MTL A-2
 RID 2-15
 rti.mtl 2-15
 RTI.RID 2-15
 filtering
 adding object 3-15
 by ID 3-15
 object 3-15
 object display 3-11
 objects A-6
 objects and interactions 3-13
 removing object from filter list 3-15
 translation of objects 3-14
 Fire PDU 1-8
 FLEXlm
 directory location 2-7
 installing 2-7
 license file 2-9

- license manager 2-6
- license server name 2-8
- multiple licenses 2-10
- troubleshooting 2-13
- FOM 2-15
 - supported 1-3
- FOM agility 1-2
- FOM Mapper 1-2, 3-8
- FOM mapper library A-7
 - initialization data A-7
- forceDisUpdates** parameter 3-12, A-7
- forceHlaUpdates** parameter 3-12, A-7
- forcing update 3-12

G

- gateway
 - log file A-5
- gateway.log file 3-17, A-4
- gateway.mtl A-4
- gateway.mtl file 3-6, 3-8, 3-17
- gatewaySendPort** parameter 3-6, A-7
- global object designator
 - mapping to DIS entity ID A-5
- global object ID
 - parsing unknown 3-16
- Gridded Data PDU 1-12
- GriddedData class 1-12
- GRIM RPR 1-3
- GroundVehicle* class 1-7
- GUI
 - configuring A-10
 - running without 3-4

H

- H startup option 3-5
- h startup option 3-5
- handle
 - mapping 1-4
- heartbeat 1-6, 1-11, A-4
- heartbeatInterval** parameter 3-6, A-4
- HLA
 - advisories A-8
 - filtering objects A-8
 - forcing updates 3-12
 - object 1-5, 3-10
 - services 1-3
- HLA 1516 1-3
- HLA ID
 - enabling or disabling display of A-9
 - filtering display of 3-11

- HLA object
 - incomplete 3-10
- HLA RPR FOM Entity ID
 - filtering by 3-15
- HLA RTI 1.3 1-3
- HLA RTI Object ID
 - filtering by 3-15
- hlaEntityIdObjectFilters** parameter A-7
- hlaExecName** parameter 3-6, A-4
- hlaFedFileName** parameter 3-6, 3-8, A-4
- hlaFedName** parameter 3-6, A-4
- hlaFomMapperLibInitData** parameter 3-8, A-7
- hlaFomMapperLibName** parameter 3-8, A-7
- hlaMaxTickTime** parameter A-7
- hlaMinTickTime** parameter A-8
- hlaObjectFilters** parameter A-8
- hlaRPRFOMVersion** parameter 3-6, 3-8, A-4
- hlaUseAdvisories** parameter A-8
- host entity ID 1-10
- host ID
 - getting 2-8
- host name 2-8
- host object ID 1-10
- HostObjectID** parameter 1-5

I

- ID
 - parsing unknown 3-16
- IFF* class 1-5, 1-10
- IFF PDU 1-11
- IffInterrogator* class 1-10, 1-11
- IffTransponder* class 1-10, 1-11
- Inactive 3-10
- inc 3-10
 - in HLA object display 3-10
- incomplete
 - object information 1-5, 3-10
- incomplete message
 - translating 3-16
- incomplete object 3-10
- installing
 - DMSO RTI 2-16
 - FLEXIm software 2-7
 - RTI 2-15
- interaction 1-7
 - ApplicationSpecificRadioSignal* 1-10
 - clearing list 3-11

- collision 1-8
- DatabaseIndexRadioSignal* 1-10
- enabling or disabling display of A-9
- EncodedAudioRadioSignal* 1-10
- filtering 3-14
- MunitionDetonation* 1-8
- number to display 3-16
- RadioSignal* 1-10
- RawBinaryRadioSignal* 1-10
- subscribing to 3-13
- translating 3-13
- viewing discovered 3-10
- viewing list of 3-10
- WeaponFire* 1-8

J

- JammerBeam* class 1-11

L

- license file 2-9
 - adding new 2-10
- license server
 - name 2-8
 - running 2-11
 - status 2-11
 - stopping 2-11
 - troubleshooting 2-13
- Linux
 - running Gateway in background 3-4
- lmdown program 2-11
- lmhostid 2-8
- lmstat program 2-11
- log file 3-17, 3-18, A-4, A-5
- logFileName** parameter A-4
- Logistics protocol family 1-12

M

- MÄK RTI
 - configuring 2-15
- MÄK Technologies LISP A-2
- MAKLMGRD_LICENSE_FILE environment
 - variable 2-9
- manual
 - conventions 1-xi
- mapping
 - entity type to object class 1-7

- marking text
 - enabling or disabling display of A-9
 - filtering display of 3-11
- memory
 - usage for interaction display 3-16
- message 3-17
 - translating incomplete 3-16
- missing
 - attributes or parameters 1-4
- MTL
 - file A-2
- MTL file 3-17
- multicast
 - subscribing to A-8
- multicast traffic A-6
- multicastSubscriptions** parameter A-8
- multicastTimeToLive** parameter A-8
- MunitionDetonation* interaction 1-8
- MunitionEntity* class 1-7

N

- n startup option 3-5
- network
 - traffic A-6
- notification level 3-17
- notify level A-5
- notifyLevel** parameter 3-17, A-5
- numberOfInteractionsShown** parameter 3-16, A-9

O

- object
 - adding to filter list 3-15
 - enabling or disabling display of A-10
 - filtering A-6
 - filtering and unfiltering 3-15
 - filtering by ID 3-15
 - filtering data display 3-11
 - filtering translation of 3-14
 - incomplete HLA 3-10
 - sorting display of 3-11
 - subscribing to 3-13
 - time out A-4
 - translating 3-13
 - unfiltering 3-15
 - updates 1-5, 3-10
 - viewing discovered 3-10
 - viewing list of 3-10
- object ID

filtering by A-8

P

-P startup option 3-5

packet

dropping A-7

parameter

autoRestart 3-18, A-9

autoStart 3-6, A-4

BeamID 1-5

BeamParameterIndex 1-5

commenting out A-2

debugEntityIdMapping A-5

debuggedTranslators 3-17, A-6

debugGlobalIdMapping A-5

Default

0.0.0.0 (the default network address.) A-7

disabledTranslators 3-14, A-6

disApplicationNum 3-6, A-4

disDrainTimeout A-6

disExerciseld 3-6, A-4

disMcastDevice A-6

disNumPdusToReadPerTimeCheck A-6

disObjectFilters A-6

displayLastUpdateThreshold A-9

disPort 3-6, A-4

disProtocolVersionToRecvMax A-6

disProtocolVersionToRecvMin A-6

disProtocolVersionToSend A-6

disSiteId 3-6, A-4

disSocketReceiveBufferSize A-6

dumpGuiToText 3-18, A-5

EmitterName 1-5

EmitterNumber 1-5

EmittingSystemID 1-5

EntityState 1-5

ERP 1-5

forceDisUpdates 3-12, A-7

forceHlaUpdates 3-12, A-7

gatewaySendPort 3-6, A-7

heartbeatInterval 3-6, A-4

hlaEntityIdObjectFilters A-7

hlaExecName 3-6, A-4

hlaFedFileName 3-6, 3-8, A-4

hlaFedName 3-6, A-4

hlaFomMapperLibInitData 3-8, A-7

hlaFomMapperLibName 3-8, A-7

hlaMaxTickTime A-7

hlaMinTickTime A-8

hlaObjectFilters A-8

hlaRPRFOMVersion 3-6, 3-8, A-4

hlaUseAdvisories A-8

HostObjectID 1-5

logFileName A-4

missing 1-4

multicastSubscriptions A-8

multicastTimeToLive A-8

notifyLevel 3-17, A-5

numberOfInteractionsShown 3-16, A-9

parseUnknownGlobalIds 3-16, A-8

pollInterval 3-6, A-4

publisherTickInterval A-8

renameDisObjects 1-5, A-8

showGUI 3-4

showGui A-9

showInteractions A-9

showMarkingText A-9

showObjects A-10

showObjectUpdates A-10

strictDiscoveryCriteria 3-16, A-8

SystemType 1-5

timeoutInterval A-4

timeStampAbsolute 3-12, A-5

useAstiRadioModes A-8

parseUnknownGlobalIds parameter 3-16, A-8

path

specifying for RTI 2-15

PDU 1-2

Acknowledge 1-9

Action Request 1-9

Action Response 1-9

Collision 1-7

Comment 1-9

Create Entity 1-9

Data 1-9

Data Query 1-9

delta state EE 1-11

designator 1-11

detonation 1-8

electromagnetic emission 1-11

Entity Management 1-12

Entity State 1-7

Event Report 1-9

fire 1-8

IFF 1-11

incomplete 1-5, 3-10

protocol families 1-6

radio transmitter 1-10

reading A-6

- Remove Entity 1-9
- Repair Complete 1-12
- Repair Response 1-12
- Resupply Cancel 1-12
- Resupply Offer 1-12
- Resupply Received 1-12
- sending and receiving 3-7, A-4
- service request 1-12
- Set Data 1-9
- simulation management (SIMAN) 1-9
- Start/Resume 1-9
- state update EE 1-11
- Stop/Freeze 1-9
- supported 1-6
- Synthetic Environment 1-12
- timeout for reading A-6
- performance A-9
 - configuring A-6
 - displaying interactions 3-16
- PhysicalEntity* class 1-8
- pollInterval** parameter 3-6, A-4
- port 3-7, A-4, A-7
- position attribute 1-8
- Preferences dialog box 3-7
- processing
 - stopping 3-12
- products
 - technical support x
 - upgrades x
- protocol
 - family 1-6
- Protocol Data Unit 1-2
- publication 1-4
- publisherTickInterval** parameter A-8
- publishing
 - incomplete objects 3-10

Q

- quitting Gateway 3-6

R

- R startup option 3-5
- RadarBeam* class 1-11
- Radio Communications protocol family 1-10
- radio object 1-10
- Radio Transmitter PDU 1-10
- RadioReceiver* class 1-5, 1-10
- RadioSignal* interaction 1-10

- RadioTransmitter* class 1-5, 1-10
- RawBinaryRadioSignal* interaction 1-10
- reading
 - PDU's A-6
- Real-Time Platform Level Reference 1-2
- receiver protocol, signal protocol 1-10
- receiving
 - PDU's 3-7, A-4
- relative timestamp 3-12
- Remove Entity PDU 1-9
- RemoveEntity* class 1-9
- removing
 - object from filter list 3-15
- renameDisObjects** parameter 1-5, A-8
- Repair Complete PDU 1-12
- Repair Response PDU 1-12
- RepairComplete* class 1-12
- RepairResponse* class 1-12
- restarting translations automatically 3-18
- Resupply Cancel PDU 1-12
- Resupply Offer PDU 1-12
- Resupply Received PDU 1-12
- ResupplyCancel* class 1-12
- ResupplyOffer* class 1-12
- ResupplyReceived* class 1-12
- RID file 2-15
- rid.mtl* file 2-15
- RPR FOM 1-2, 3-8, A-4
 - filtering by entity ID attribute A-7
- RTI 2-15
 - definition of 2-15
 - DMSO, installing 2-16
 - environment 3-3
 - losing connection to 3-18
 - ticking A-7
- RTI advisory A-8
- RTI_CONFIG environment variable 2-15
- rtiexec 2-16, 3-3
- RTI.rid* file 2-15
- runLm program 2-11
- running
 - Gateway 3-10
 - license server 2-11

S

- S startup option 3-5
- S startup option. 3-5
- s startup option 3-5
- sending

- PDU 3-7, A-4
- Service Request PDU 1-12
- ServiceRequest* class 1-12
- Set Data PDU 1-9
- SetData* class 1-9
- showGUI** parameter 3-4
- showGui** parameter A-9
- showInteractions** parameter A-9
- showMarkingText** parameter A-9
- showObjects** parameter A-10
- showObjectUpdates** parameter A-10
- shutting down
 - license server 2-11
- SIMAN PDU 1-9
- Simulation Management PDU 1-9
- SISO 1-3
- SISO DLC 1516 1-3
- site ID 3-7, A-4
- socket
 - buffer receive size A-6
- SOM 1-4
- sorting
 - object display 3-11
- starting
 - translations 3-10
 - translations automatically 3-5
- starting gateway
 - UNIX 3-3
 - Windows 3-4
- StartResume* class 1-9
- Start/Resume PDU 1-9
- startup
 - automatic A-4
- startup option 3-6
- startup options 3-5
- State Update EE PDU 1-11
- status
 - license server 2-11
- status indicator 3-10
- Stop 1-9
- StopFreeze* class 1-9
- Stop/Freeze PDU 1-9
- stopping
 - license server 2-11
 - PDU processing 3-12
- strictDiscoveryCriteria** parameter 3-16, A-8
- subscribing
 - objects and interactions 3-13
 - to multicast addresses A-8
- subscription 1-4

- support
 - technical x
- supported PDUs 1-6
- Synthetic Environment PDU family 1-12
- SystemType parameter 1-5

T

- T startup option 3-5
- technical support x
- text
 - dumping to file 3-18
- tick A-7
- tick interval A-8
- time out A-4
- timeout 1-6
 - reading PDUs A-6
- timeoutInterval** parameter A-4
- timestamp 3-12
- timeStampAbsolute** parameter 3-12, A-5
- Transfer Control Request PDU 1-12
- TransferControl class 1-12
- translating
 - incomplete messages 3-16
 - objects and interactions 3-13
 - PDUs 3-10
- translating objects
 - filtering 3-14
- translation
 - debugging 3-13
 - restarting automatically 3-18
 - stopping 3-12
- Translation Manager 3-13, 3-17
- translations
 - starting automatically 3-5
- translator
 - debugging by default 3-17
 - disabling by default 3-14
- translators
 - disabling at startup A-6
 - enabling debug at startup A-6
- transmitter protocol 1-10
- troubleshooting
 - FLEXIm 2-13

U

- unfilter
 - object 3-15
- unknown ID

- parsing 3-16
- update
 - filtering display of 3-11
 - forcing 3-12
 - HLA object 1-5, 3-10
- updates
 - time between A-9
- upgrades x
- useAstiRadioModes** parameter A-8
- using
 - RTI advisories A-8

V

- variable
 - environment 2-9
- version
 - DIS A-6
- viewing
 - discovered objects and interactions 3-10
 - objects and interactions 3-10
- VR-Link 1-2
- VR-Link.fed 1-3

W

- warfare protocol family 1-8
- WeaponFire* interaction 1-8

X-Y-Z

- x startup option 3-5
- XML
 - RTI configuration file 2-15



Link - Simulate - Visualize

GWY-4.1.4-1-050811