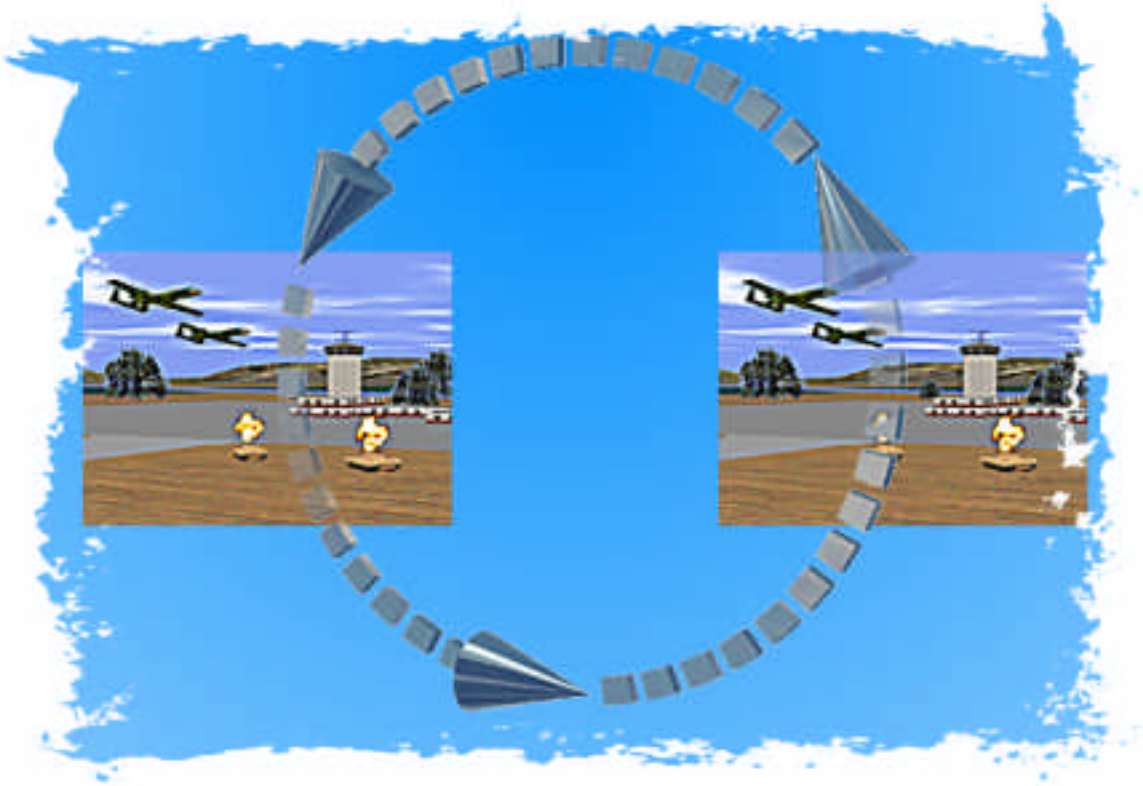
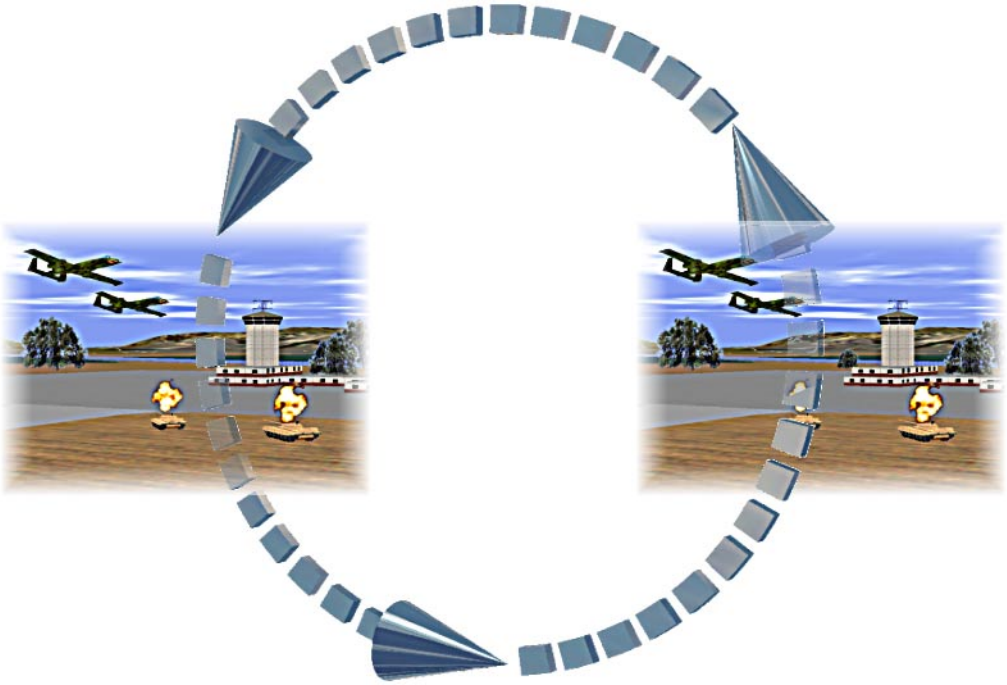


MÄK TECHNOLOGIES

# Gateway

## User's Guide



MÄK TECHNOLOGIES

# Gateway

## User's Guide

**MÄK Technologies**
185 Alewife Brook Parkway
Cambridge, MA 02138 USA

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

info@mak.com

www.mak.com

Revision GWY-3.4-1-000612

# Contents

# Preface

The MÄK Gateway is based on the DIS-HLA Gateway developed by Institute for Simulation and Training for STRICOM. It allows DIS simulations to interoperate with HLA federations that use the Real-Time Platform Level Reference (RPR) FOM.

## About this Manual

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, *Configuring the Gateway*, explains how to edit configuration files for your site.

Chapter 4, *Operating the Gateway*, explains how to start the Gateway and run a basic session. It also describes the Gateway Graphical User Interface, an alternative method of operating the Gateway.

Chapter 5, *Gateway Commands*, describes the commands you use to run the Gateway from the command line.

Appendix A, *Configuration Files*, lists the configuration files as provided by MÄK Technologies.

Appendix B, *References* is a bibliography of papers and articles describing DIS and HLA translation issues.

## Other MÄK Products

Gateway is a member of the MÄK Technologies line of HLA/DIS software products designed to streamline the process of developing and using networked simulated environments.

In addition to Gateway, the MÄK Technologies product line includes:

- The VR-Link™ Network Toolkit. The Toolkit 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/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.

- The MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK RTI is the first commercial run-time infrastructure for HLA simulations. It is optimized for real-time simulations.
- The MÄK Stealth. 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. The Stealth runs on PCs and the SGI computers and supports multi-channel views.
- The 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.
- MÄK VR-Forces. VR-Forces is a computer generated forces application and toolkit. It provides an application with a graphical user interface, based on the MÄK Plan View Display, that gives you a two dimensional view of a terrain database.

  As with the Plan View Display, you can view remote entities, but you can also create your own local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.

- The MÄK Logger. The Logger, also called the Data 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 lets you choose between a graphical interface, or a shell-style, command-line interface. You can also control the Logger programmatically with logger control PDUs or state update messages.

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

Please contact your MÄK salesperson for information about these products.

## How to Contact Us

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

| | |
|---|---|
| For sales and upgrade information by e-mail: | info@mak.com |
| For technical support by e-mail: | support@mak.com |
| For license key files and information: | www.mak.com/support/keyintro.htm |
| The MÄK web site: | www.mak.com |

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

| | | |
|---|---|---|
| Send postal correspondence to: | | MÄK Technologies<br>185 Alewife Brook Parkway<br>Cambridge, MA, USA 02138 |
| Call or fax us at: | Voice: | 617-876-8085 (extension 3 for support) |
| | Fax: | 617-876-9208 |

## Document Conventions

This document uses the following typographic conventions:

| | |
|---|---|
| `Monospaced Type` | Indicates commands or values you enter. |
| *`Monospaced Italic`* | Indicates variables that you replace with appropriate values. |
| `[ ]` | 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. |
| **Bold type** | Indicates a new term. |
| *Italic* | Indicates a file name, pathname, or a class name. |
| sansSerif | 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:<br>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. |
|  | Indicates information that you must observe to prevent damage to the program. |

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

### 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.

## 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 (2.0.4, IEEE 1278.1, IEEE 1278.1a) 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.

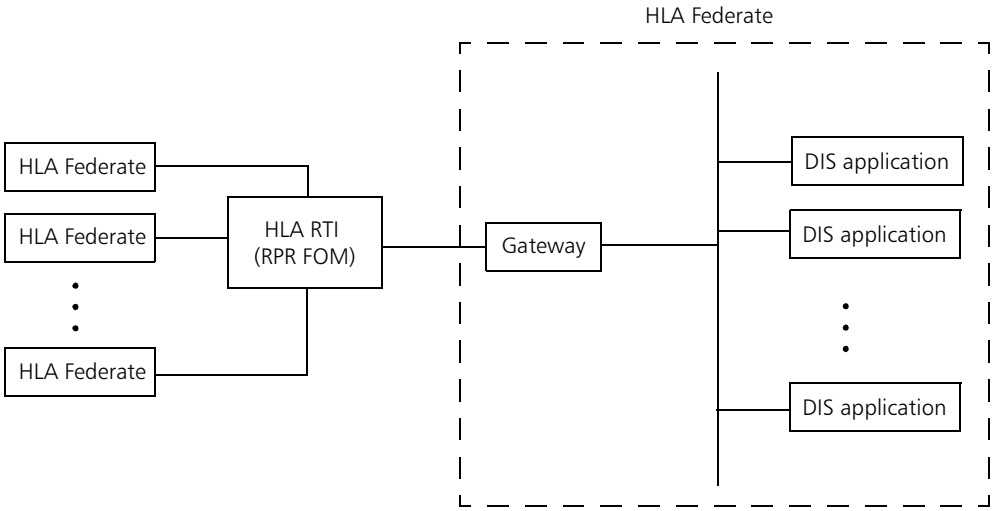



Figure 1-1. Gateway interoperability

On the DIS side of the Gateway, data packets called Protocol Data Units (PDUs) are formatted, sent, and received. The Gateway receives the packets and translates them at two levels: (1) the data in the packets is converted into the data formats defined in the RPR FOM, and (2) the sequence of packets is translated into corresponding RTI service invocations. The Gateway performs a similar conversion for data received from the HLA federation execution. Table 1-1 lists translation support for the various types of PDUs.

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. For more information, see "HLA Services," on page 1-12.

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 1-7 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.

Table 1-1: DIS Translation Capabilities

| DIS PDU Family | Support |
|---|---|
| Entity Information/Interaction | Full |
| Warfare | Full |
| Simulation Management | Full |
| Radio Communications | Full |
| Distributed Emission Regeneration | Full |
| Logistics | Full |
| 1278.1A IFF | Partial |

## 1.2. Gateway FOM Requirements

The Gateway can operate in federations that use a FOM based on RPR FOM 1.0. That is, the Gateway has the knowledge necessary to translate data between DIS and RPR FOM 1.0 representations. 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.

You can use a FED file other than VR-Link.fed. Your FED file can have classes, attributes or parameters that are not part of the RPR FOM, but the Gateway will not be able to translate the additional data. Your FOM can omit classes, attributes, or parameters, but the Gateway will not be able to translate the missing data. For example, if your FOM does not include the *WeaponFire* class, then Fire PDUs will not be translated to the HLA side. However, make sure your omissions make sense; for example, a FOM that contains the *EmitterBeam* class, but omits the *EmitterSystem* class will be problematic, since *EmitterBeams* reference their parent *EmitterSystem* objects.

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 federate's SOM will be a subset of the FOM described by VR-Link.fed. 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 to subscribe to (and thus translate) Collision interactions, meaning that Collision interactions will not be part of your federate's SOM.

### 1.2.1. Handle Mapping

The RTI identifies classes, attributes, and parameters using handles (integer identifiers). 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 any 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 not found in the FED file, the Gateway marks it as missing. The Gateway still processes the translation of the DIS data into the missing attributes. However, the missing attributes are not submitted to any of the RTI Object Management service calls. Of course, if the attribute or parameter is not defined in the FED file, it is impossible for the Gateway to receive the attribute or parameter from the RTI via the federation execution. An attribute that the Gateway requires for PDU transmission must not be omitted from the FOM.

### 1.2.2. Mandatory Attributes

The Gateway requires several attributes for creating specific DIS PDUs from HLA data. Table 1-2 lists the attributes, the containing class, and comments on any indirect requirements.

Table 1-2: Mandatory Attributes

| Class | Attribute / Parameter | Comment |
|---|---|---|
| *EmitterSystem* | 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. |
| *EmitterBeam* | 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. |
| *Designator* | HostObjectID | HostObjectID (designating entity) must refer to an object that has been registered or discovered by the Gateway. |
| *RadioTransmitter* | HostObjectID | HostObjectID (transmitting entity) must refer to an object that has been registered or discovered by the Gateway. |
| *RadioReceiver* | HostObjectID | HostObjectID (receiving entity) must refer to an object that has been registered or discovered by the Gateway. |
| *IFF* | HostObjectID, SystemType | HostObjectID (emitting/receiving entity) must refer to an object that has been registered or discovered by the Gateway. |
| *BaseEntity* | EntityID, EntityType, Position | |

Table 1-2: Mandatory Attributes

| Class | Attribute / Parameter | Comment |
|---|---|---|
| *ResupplyCancel* | ReceivingObject, SupplyingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |
| *ResupplyOffer* | ReceivingObject, SupplyingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |
| *ResupplyReceived* | ReceivingObject, SupplyingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |
| *ServiceRequest* | RequestingObject, ServicingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |
| *RepairComplete* | ReceivingObject, RepairingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |
| *RepairResponse* | ReceivingObject, RepairingObject | Objects IDs must refer to an object that has been registered or discovered by the Gateway. |

## 1.3. Translating Between DIS and the RPR FOM

This section provides an overview of the translation between DIS PDUs and RPR FOM data. The Gateway publishes at the RPR FOM leaf node classes only. As a result, all data translated to HLA will be in the form of an RPR FOM leaf node class. The Gateway can subscribe to all of the classes in the RPR FOM that permit subscription. This allows the Gateway to receive HLA data that is published from any publishable class in the RPR FOM, or one of its ancestors.

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 (see threshold settings in sections "Simulation Loading," on page 3-4 and "Entering Gateway Startup Commands," on page 4-3).
- The Gateway deletes HLA objects when the time between DIS PDUs exceeds the timeout threshold (2.5*heartbeat). RPR FOM embedded system objects (radio transmitter/receiver and emitter system/beams) are deleted only if their host entity no longer exists.

For convenience, DIS PDUs are categorized into six major protocol families. Translation between DIS and HLA is described by family, as follows:

- Entity Information and Entity Interaction PDUs
- Warfare PDUs
- Simulation Management PDUs
- Radio Communications PDUs
- Distributed Emission Regeneration PDUs
- Logistics PDUs.

### 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 1-3. 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 Gateway configuration file provides a mechanism for specifying alternative mappings (see Chapter 3, *Gateway Configuration*). 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 1-3: 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 1-4.

Table 1-4: 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 1-5. 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 1-5: 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 1-6: 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 1-5). 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 *EmittingSystemID* 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 1-7.

Table 1-7: Logistics Representation in the RPR FOM

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

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

## *1.4. HLA Services*

In general, the HLA Federation Management and Declaration Management services have no corresponding DIS functionality. The Gateway must perform these services on behalf of the DIS simulation exercise (via user commands). The HLA Object Management services are handled automatically by the Gateway's translation functionality. Chapter 4, *Operating the Gateway*, and Chapter 5, *Gateway Commands*, describe how the Gateway's commands and operation support the HLA services.

# 2

# Installing the Gateway

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

# 2.1. Installing the Gateway

To use the Gateway, you need to install the Gateway software and a supported RTI.

## 2.1.1. Installing the Gateway on SGI IRIX or Linux

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

### Installing the Gateway from CD-ROM

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

To install the Gateway:

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

   ```
   ./install
   ```

3. Follow the on-screen instructions. You can install the Gateway into any directory for which you have write permissions.
4. Copy the RTI federation file *VR-Link.fed* to the RTI configuration directory (*$RTI_HOME/config*). See RTI documentation for information about the RTI_HOME environment variable.

### Installing the Gateway from A Downloaded File

If you obtained the Gateway via ftp, you received a compressed tar file. To install the Gateway:

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

   ```
   gunzip gateway.tar.gz
   ```

4. Untar the file

   ```
   tar -xvf gateway.tar
   ```

5. If you are using the DMSO RTI, copy *VR-Link.fed* to the RTI configuration directory (*$RTI_HOME/config*). See RTI documentation for information about the RTI_HOME environment variable.

## 2.1.2. Installing the Gateway On A PC

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

Before you install the Gateway, read the *Release Notes* to see if there are any special instructions for installation. You need to know which version of the RTI you plan to use.

### Installing the Gateway from A CD

To install the Gateway from a CD:

1. Insert the Gateway CD into your CD-ROM drive. If the autorun feature is enabled, the MÄK PC Products Installation screen opens.

   If the MÄK PC Products Installation does not open, open the Windows Explorer. In the root directory for your CD drive, double-click *makInst.exe*. The MÄK PC Products Installation screen opens.

2. In the MÄK PC Products Installation screen, click Gateway or Gateway-ngc, depending on which version of the RTI you are using. (You can install both versions if you have enough disk space. Both versions support DIS.)

3. Follow the instructions of the installation program.

### Installing the Gateway from A Downloaded File

The Gateway is distributed via ftp as a zip file.

To install the Gateway from a zip file:

1. Create the directory into which you want to install the Gateway.

2. Copy the zip file to a temporary directory.

3. Unzip the zip file.

4. Run *setup.exe*.

5. Follow the instructions of the installation program.

### 2.1.3. Directory Structure

The directory structure for the installed files is:

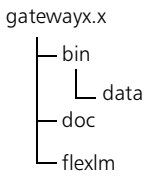



Figure 2-1. Gateway directory structure

Table 2-1 describes the contents of the Gateway directories.

Table 2-1: Gateway directory contents

| Directory | Contents |
|---|---|
| bin | Executables, FED files, RID files, and configuration files. |
| data | Files required for the optional Gateway GUI. |
| doc | Gateway documentation. |
| flexlm | License management files. |

## 2.2. 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.

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.

The two RTI implementations against which all MÄK products have been tested are the MÄK RTI (which comes with all MÄK products), and the DMSO RTI (which is available from the DMSO web site). For the most recent information about the RTI versions supported by the Gateway, see the Product Versions page on the MÄK Technologies web site at: http://www.mak.com.

### 2.2.1. Installing the MÄK RTI

The MÄK RTI is included with all MÄK products and you have the option of installing it as part of product installation.

#### *UNIX*

The installation process creates a link called *RTI* in each product's root directory, which points to the *makRti* directory, in which the MÄK RTI software is located.

#### *PCs*

The project files point to the default installation directory for the RTI.

#### Configuring Your System to Use the MÄK RTI

The RTI dynamic library needs to be located somewhere on your dynamic library search path. This path is indicated by an environment variable as follows:

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

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

- The FED file (*.fed*) 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

You do not need an RTI server, such as the *rtiexec*, or central application to use the MÄK RTI, however you can run the *rtiexec* if you want to. 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 the *MÄK RTI Reference Manual* for information about the *rtiexec* and changing the networking parameters such as multicast addresses and UDP ports.

#### Documentation for the MÄK RTI

The documentation for the MÄK RTI consists of:

- *MÄK RTI Reference Manual.*
- The documentation set for the DMSO RTI. Aside from the installation and configuration instructions, the documentation for the DMSO RTI also applies to the MÄK RTI. It is included on the MÄK product CD.

### 2.2.2. Installing the DMSO RTI

You can get the DMSO RTI from DMSO's web site (http://hla.dmso.mil). Click on Software Distribution Center, where you can register for an account, then later download the RTI software. Documentation for the DMSO RTI is provided on the MÄK product CD because the documentation applies to the MÄK RTI as well.

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

1. Install the RTI according to the instructions provided.
2. Copy the *giant.fed* file from the MÄK products root directory to the DMSO RTI's config directory.

#### 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 resides in the RTI's *bin/platform_name* directory (UNIX) or in the RTI's *bin* directory (PCs). Run *rtiexec* only on one machine, regardless of how many HLA applications you are running.

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

## 2.3. Setting Up and Using the FLEXlm License Manager

The Gateway, like other MÄK Technologies products, uses the FLEXlm license manager. Before you can use the Gateway, you must obtain a valid license file and configure the license server and client machines. (For information about FLEXlm, view the FAQ page at www.globetrotter.com/flexlm.htm.) 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 executable, *maklmgrd* 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 has a license file (supplied by MÄK Technologies). This file identifies the MÄK Technologies 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 more machines than you have licenses for, 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. Then, you must update the MAKLMGRD_LICENSE_FILE environment variable on every client. This is not a trivial process, and is not recommended. Keep this in mind when you choose the machine to use as a license server.

---

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

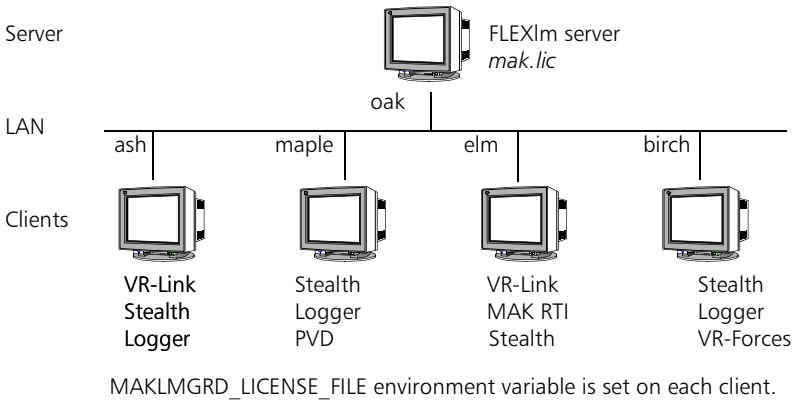



Figure 2-2. FLEXlm example architecture

## 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 2-13.

### Installing the License Manager Software

As part of the installation of all MÄK products, the FLEXlm software is installed in the *flexlm* directory under the root product directory.

1. Install your MÄK products on the machine(s) on which you want to run them.
2. If this is a networked system, and you did not install MÄK software on the license server, copy the *flexlm* directory from a client machine to the license server machine.

### Requesting A License File

License files are keyed to the host id of the license server. You need to supply MÄK with some information so that we can generate a license file for you.

To get a license file:

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

   On Windows platforms, execute *lmhostid.bat*. A command window opens and displays the host ID. Write down the number.
3. Go to http://www.mak.com/support/keyintro.htm and complete the license key request form. MÄK will e-mail you a file named *mak.lic*.

To get a license file, you need to know the name of the license server machine. To get the name of your machine follow the procedure for your operating system:

#### *UNIX*

At a command prompt, type `hostname`.

#### *Windows 95/98/NT*

1. On the Start menu, choose Settings → Control Panel.
2. Open the Network applet.
3. Select the Identification tab. The machine name is listed in the Computer Name field.

### Putting the License File in the *flexlm* Directory

After you receive the license file, put it 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 2-13.

### Setting the MAKLMGRD_LICENSE_FILE Environment 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 95/98*

On Windows 95/98, you set environment variables by adding or editing lines in the *Autoexec.bat* file. To set the MAKLMGRD_LICENSE_FILE variable, add a line similar to the following:

```
set MAKLMGRD_LICENSE_FILE=@oak
```

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

#### *Windows NT*

On Windows NT, you set environment variables in the Environment tab of the Properties dialog box for the machine:

1. Right-click the My Computer icon on the Desktop.
2. On the pop-up menu, select Properties. The System Properties dialog box opens (Figure 2-3).
3. Select the Environment tab.
4. In the Variable text field, type MAKLMGRD_LICENSE_FILE.
5. In the Value field, type the value, for example:

   `@oak`

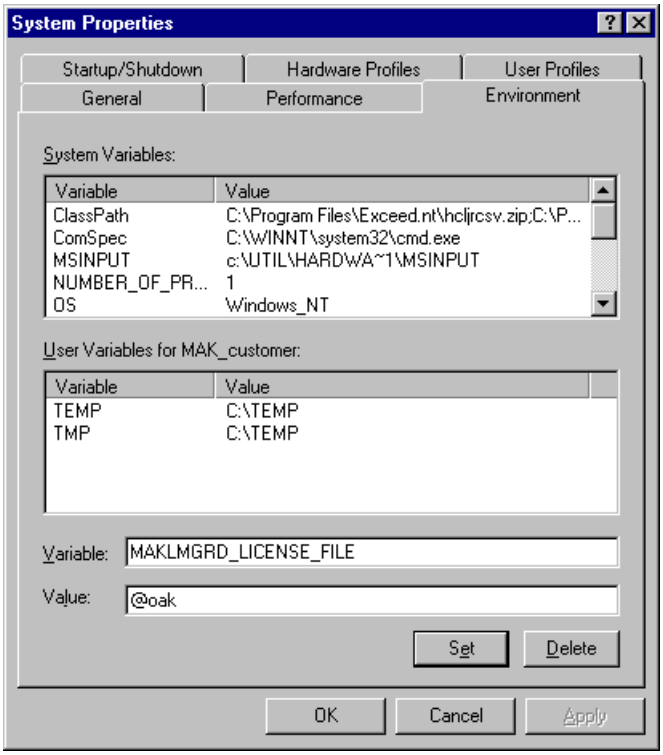



Figure 2-3. Setting an environment variable in Windows NT

### *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 2-13.

### 2.3.3. 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 2-10.

To add a new license file to your license server:

1. When you get the new license file, rename it from *mak.lic* to something like *mak2.lic*, or some other name ending in *.lic*, so that it has a different name from the license file(s) 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.

### 2.3.4. 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.

> To obtain the current status of the server, run *lmstat*.

---

! 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*.

---

#### Shutting Down the License Server

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

## 2.3.5. Troubleshooting the FLEXlm License Manager

For general troubleshooting information, refer to the FLEXlm FAQ at www.globetrotter.com/flexlm.htm. 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 95/98/NT, check the Task Manager. To open the Task Manager, press Ctrl-Alt-Delete.

   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 NT, select a process in the Task Manager Processes window and click End Process.

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

3. 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:\Program Files\MAK\flexlm\maklmgrd"
   ```

---

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

---

# 3

# Configuring the Gateway

This chapter describes the keywords in Gateway configuration files. The text of the files is in Appendix A, *Configuration Files*.

## 3.1. Gateway Configuration

The Gateway requires certain configuration and data files to be present in the directory from which you run the executable. These files are provided as part of this release with default values. The configuration files are as follows:

- *Sim.cfg* Contains general configuration data
- *Sim.lod* Contains load management data
- *Gateway.cfg* Contains federate configuration data.

You must set some configuration options before you can use the Gateway. The configuration files that require editing are *Sim.cfg*, and *Sim.lod*. The options that you must set are:

- SITE
- HOST
- EXERCISE
- NET_MASK
- IP_ADDRESS
- UDP_PORTS.

---

**!** If you edit a configuration file, do not insert any control characters, except tab and end-of-line.

---

Configuration file entries use the keyword = value format. Spaces are ignored. A configuration entry consists of a single line, and only one entry is allowed per line. You can end a line with a comment or put a comment on its own line. Precede a comment with an asterisk (*). The following lines are examples of configuration file entries:

```
SITE = 11 * This entry is the HLA Gateway site.
* This comment is on a single separate line.
```

Keywords may be in upper, lower, or mixed case, for example, `SITE`, `site`, or `Site`).

---

**!** Do not change any configuration parameters other than those described in this chapter. The parameters that are not described are functionally obsolete, but removing them may cause the Gateway to run incorrectly.

---

### 3.1.1. General Configuration

The configuration file *SIM.CFG* specifies network and connectivity information. Some keywords are optional; others are mandatory. If you do not specify an optional keyword, it is initialized with a default value. Table 3-1 describes each keyword.

Table 3-1: Configuration file keywords

| Keyword | Description |
|---|---|
| | **Required keywords** |
| SITE | Specifies the site component of the Gateway address and the DIS entity identification record. The site value cannot be zero, $2^{16}$-1, or $2^{16}$-2 (see the DIS Standard for further details). Each site must have a unique site number.<br>Example: `site = 12` |
| HOST | Specifies the host component of the Gateway address and the DIS entity identification record. These values must be unique across all gateway applications at any given site (see the DIS Standard for further details).<br>Example: `host = 88` |
| EXERCISE | Specifies the DIS simulation exercise.<br>Example: `exercise = 1` |
| NET_MASK | Specifies the broadcast mask for the DIS network communications. This string must be enclosed in quotation marks.<br>Example: `net_mask = "255.255.255.0"` |
| IP_ADDRESS | Specifies the internet protocol (IP) address of the machine that is running the Gateway. This string must be enclosed in quotation marks.<br>Example: `ip_address = "123.107.119.88"` |
| UDP_PORTS | Specifies the UDP ports for DIS, Output, Gateway Send Message, and Gateway Receive Message:<br>`udp_ports="DIS,Output,Send Message,Receive Message"`<br>This string must be enclosed in quotation marks.<br>Note: The receive message and send message ports are used by the GUI (see "Managing Profiles," on page 4-10).<br>Example: `udp_ports = "3000,3001,3002,3003"` |
| | **Optional Keywords** |
| GW_CFG | Specifies the name of the gateway configuration file. The value must be in quotes.<br>Example: `gw_cfg = "gateway.cfg"` |

Table 3-1: Configuration file keywords

| Keyword | Description |
|---|---|
| NET_INTERFACE | If more than one network card is present, specifies which network interface card to use for the DIS network, for example, ec0, ec2, and so on, on Indys.<br>Example: `net_interface = "ec2"` |
| DIS_BIG_ENDIAN | Specifies that data sent or received via a DIS network is big-endian or little endian. Values are yes or no.<br>Default: Yes |
| HLA_BIG_ENDIAN | Specifies that data sent or received via an HLA network is big-endian or little endian. Values are yes or no.<br>Default: Yes |
| RESIGN_AT _EXIT | Specifies whether or not to resign from a federation automatically when the federate exits. Values are yes or no<br>Default: Yes.<br>RTI NG does not support this keyword. |

## 3.1.2. Simulation Loading

The configuration file, *SIM.LOD*, controls load management and appearance. For example, in the load file you can:

- Specify display options
- Change the DIS network interface.

Although the file must exist, all keywords in this file are optional. If a keyword is not specified it is initialized with a default value. Table 3-2 describes each keyword and its default value.

Table 3-2: SIM.LOD configuration keywords

| Keyword | Description |
|---|---|
| MAX_TICK_TIME | Specifies the maximum time for the RTI to process events.<br>Default: 0 (No limit.) |
| MAX_PACKETS_ PER_CYCLE | Specifies the maximum number of packets per cycle.<br>Default: 0 (No limit.) |

Table 3-2: SIM.LOD configuration keywords

| | |
|---|---|
| HEARTBEAT_ INTERVAL | Specifies the heartbeat, gateway update interval, heartbeat tolerance and timeout factor for Designator, Entity-state, Transmitter, Receiver, Emission, and IFF PDUs.<br>By default, the Gateway checks for heartbeat and timeout thresholds at a rate specified by the minimum interval value. The gateway update rate can be increased above this rate, but cannot be decreased below it.<br>Default values<br>PDU heartbeats - 5.0 seconds<br>time out factor - 2.5 seconds<br>heartbeat tolerance - 1.1.<br>For more information, see "Specifying the Heartbeat Interval," on page 3-7. |

Table 3-2: SIM.LOD configuration keywords

| | |
|---|---|
| PDU_KIND | Assign an alternative PDU version, family, and kind to the specified PDU type. This option allows the Gateway to send and receive specific DIS PDUs with different versions from the Gateway executable (for example, send/receive a 1278.1A IFF PDU with a DIS 2.0.4 Gateway).<br>It can also be used to designate PDU kind and family values other than those defined in the enumeration document (for example, send/receive the IFF PDU with the experimental family and kind values). Received DIS PDU kind values are mapped to the PDU type specified in the entry and compared against the entry's version (non-matching causes the PDU to be dropped). Sent DIS PDUs take on the specified version, family, and kind. The keywords allowed are:<br>◆ ACKNOWLEDGE<br>◆ ACTION_REQUEST<br>◆ ACTION_RESPONSE<br>◆ COLLISION<br>◆ CREATE_ENTITY<br>◆ DATA<br>◆ DATA_QUERY<br>◆ DESIGNATOR<br>◆ DETONATION<br>◆ EMISSION<br>◆ ENTITY_STATE<br>◆ EVENT_REPORT<br>◆ FIRE<br>◆ IFF<br>◆ MESSAGE<br>◆ RADIO_RECEIVER<br>◆ RADIO_SIGNAL<br>◆ RADIO_TRANSMITTER<br>◆ RESUPPLY_CANCEL<br>◆ RESUPPLY_OFFER<br>◆ RESUPPLY_RECEIVED<br>◆ SERVICE_REQUEST<br>◆ REPAIR_COMPLETE<br>◆ REPAIR_RESPONSE<br>◆ REMOVE_ENTITIY<br>◆ SET_DATE<br>◆ START_RESUME<br>◆ STOP_FREEZE<br>The sending functionality is implemented only for the IFF PDU.<br>Example: `PDU_KIND = (ENTITY_STATE = (4 1 1) IFF = (6 6 28))` |

Table 3-2: SIM.LOD configuration keywords

| | |
|---|---|
| INPUT_ WINDOW | The input_window value, given as Yes or No, specifies whether the input window is enabled or disabled. When disabled, the input window is not displayed. Default: Yes.<br>If the input window is disabled, window management is not initialized. This configuration reduces the general overhead of the Gateway.<br>Example: `input_window = no` |
| KEYBOARD | Specifies whether keyboard input is enabled or disabled. This is a legacy command with no current use, and should always be set to yes. The default value is yes and the ESC key is always enabled.<br>Example: `keyboard = yes` |

### Specifying the Heartbeat Interval

PDUs are sent out at the interval determined by the formula:

```
heartbeat_interval * heartbeat_tolerance
```

When a DIS simulator fails to send out PDUs at the appropriate heartbeat rate, the gateway times out (that is, deletes) the associated federation object. The timeout value is determined by the formula:

```
heartbeat_interval * timeout_factor
```

The syntax for the heartbeat interval is:

```
Heartbeat_Interval =
(
(designator value)
(entity_state value)
(transmitter value)
(receiver value)
(emission value)
(gateway value)
(iff value)
(to_factor value)
(hb_factor value)
)
```

where *value* is a floating point value

The **Heartbeat_Interval** keyword is optional and each value is optional.

## 3.1.3. Gateway Configuration

The Gateway configuration file specifies the default values used by the Gateway for RTI Federation Management and Declaration Management. The values of the Declaration Management options are formatted as lists of strings. The strings must be sub-strings of the fully qualified FOM class names, such that the complete class name is used for each class level (for example, *PhysicalEntity.Platform*). If a fully qualified FOM class name is found to contain the string, that FOM class is added to the corresponding option's class list. Object and interaction class names are not cross-matched.

Table 3-3 describes each keyword and its default value.

Table 3-3: Gateway configuration keywords

| Keyword | Description |
|---|---|
| FEDERATE_NAME | Specifies the default federate name. If this keyword is not specified, the default federate name is *gateway*.<br>Example: `FEDERATE_NAME = "gateway"` |
| FEDERATION_NAME | Specifies the default federation name. If this keyword is not specified the default federation name is *VR-Link*.<br>Example: `FEDERATION_NAME = "vr-link"` |
| FED_FILE | Specifies the default FED file name. If this keyword is not specified the default FED file name is *vr-link.fed*.<br>Example: `FED_FILE = "vr-link.fed"` |
| VERBOSE_OUTPUT | Specifies verbose debugging output. Default: no.<br>Example: `VERBOSE_OUTPUT = no` |
| BUILD_OBJECTNAMES | Specifies that object names are built from DIS Entity ID data. Default: yes<br>Example: `BUILD_OBJECTNAMES = yes` |
| SUBSCRIBE_INTERACTIONS | Overrides the default interaction subscription with this list of interaction class names. The default interaction subscription list is the complete set of subscribable interaction class names.<br>Example: `SUBSCRIBE_INTERACTIONS = ( "Detonation" "WeaponFire" "Collision" )` |
| PUBLISH_INTERACTIONS | Overrides the default interaction publication with this list of interaction class names. The default interaction publication list is the complete set of publishable interaction class names.<br>Example: `PUBLISH_INTERACTIONS = ( "Detonation" "WeaponFire" "Collision" )` |

Table 3-3: Gateway configuration keywords

| Keyword | Description |
|---|---|
| SUBSCRIBE_OBJECTS | Overrides the default object subscription with this list of object class names. The default object subscription list is the complete set of subscribable object class names.<br>Example: `SUBSCRIBE_OBJECTS = ( "BaseEntity" )` |
| PUBLISH_OBJECTS | Overrides the default object publication with this list of object class names. The default object publication list is the complete set of publishable object class names.<br>Example: `PUBLISH_OBJECTS = ( "BaseEntity" )` |
| ENTITY_TYPE_MAPPING | Overrides the default mapping between DIS entity types and RPR FOM object classes. The default mapping is based on the entity type kind and domain fields. The format of this option is a list of assignments where the identifiers are fully qualified object class names that are derived from *BaseEntity*. The value of each object class assignment is a list containing integer lists. Each integer list represents an entity type where the position of the integer is associated with a field in the entity type structure as follows: (kind domain country category sub-category specific extra).<br>Example:<br>`ENTITY_TYPE_MAPPING =`<br>`(`<br>`* a flying M1`<br>`BaseEntity.PhysicalEntity.MilitaryEntity. Platform.MilitaryAirLandPlatform = ( (1 1 225 1 1 0 0) )`<br>`* a grounded A10`<br>`BaseEntity.PhysicalEntity.MilitaryEntity. Platform.MilitaryLandPlatform = ( (1 2 225 2 4 0 0) )`<br>`)` |

# 4

# Operating the Gateway

This chapter explains how to start the Gateway and enter commands.

# 4.1. Running the Gateway

Operating the Gateway involves the following general steps:

1. If you are using a DMSO RTI, start the rtiexec.
2. Start the appropriate Gateway executable.
3. Enter commands in the Gateway Command Input window.
4. Optionally, use the Gateway Graphical User Interface (GUI) to send commands.

These steps are described in the remainder of this chapter.

If the DIS exercise that you want to translate has multiple DIS applications, we recommend that you do not run an instance of the Gateway for each DIS application. Use one Gateway for all the DIS applications. If you must run multiple Gateways, be sure that they use separate DIS ports or exercise IDs.

## 4.1.1. Setting Up the RTI Environment

To use the Gateway you must set up an RTI environment. See "Installing An RTI," on page 2-5 for information about setting up your environment with either the MÄK RTI or DMSO RTI.

> If you are using DMSO RTI 1.3vx or RTI-NG, start the rtiexec.

A command window opens with text similar to the following:

```
rtiexec@mycomputer>
Discovery request issued.
```

## 4.1.2. Starting the Gateway

To start the Gateway:

1. In a command shell, change directory to the *gateway/bin* directory.
2. Enter the name of the executable that you want to run:

   `gateway_XX[NG]`

   where *XX* specifies the DIS version being translated to HLA (for example GATEWAY_204_13.EXE for DIS 2.0.4), and NG indicates support for the DMSO RTI-NG or MÄK RTI -ngc. See the release notes for a list of the executables provided with your version of the Gateway.

   The command window displays a series of messages, and the Gateway Command Input window opens (Figure 4-1).

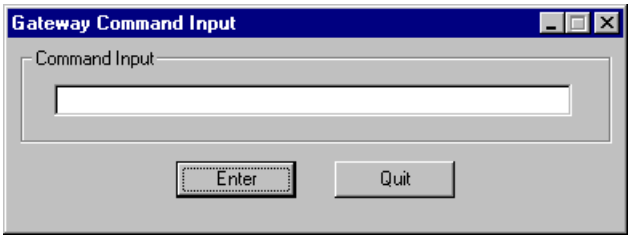



Figure 4-1. Gateway Command Input window

If you are running the Gateway on a PC, you can create shortcut icons for the different executables and start the Gateway by clicking on them, rather than typing the executable name.

### The Gateway Command Input Window

When the Gateway starts, it displays the Gateway Command Input window (Figure 4-1). You use the Gateway Command Input window to enter Gateway commands. The command shell window displays system output. You can disable either or both of these windows (see "Simulation Loading," on page 3-4). Command input is received from the command shell if the Gateway Command Input window is disabled.

You can enter commands in the Gateway Command Input window or in the command shell from which you started the Gateway. However, if you use the command shell to enter commands, the text that you enter may scroll off the window as output is displayed. You can continue to enter command text, but the effect may be disconcerting. You can use scripts to mitigate this problem.

## 4.1.3. Entering Gateway Startup Commands

The Gateway commands are described in detail in Chapter 5, *Gateway Commands*. A typical Gateway session consists of the following sequence of actions:

1. Autostart the Gateway connection to the VR-Link federation execution. The VR-Link federation execution is created if it does not already exist. In the Command Input text box in the Gateway Command Input window, enter the command:

   ```
   h b
   ```

2. Press the Enter key, or click the Enter button.
3. Activate the RTI and PDU interfaces. Enter the command:

   ```
   h t
   ```

   The Gateway translates data between DIS and the VR-Link federation execution.

### 4.1.4. Closing the Gateway

Before you close the Gateway, you should resign from the federation.

> To resign from the federation, in the Gateway Command Input window, enter the command:

```
h r
```

> To quit the Gateway, in the Gateway Command Input window, click the Quit button.

## 4.2. Troubleshooting the Gateway

The Gateway directs all output to the terminal window from which the Gateway executable was started.

**Problem:** The Gateway is not receiving DIS PDUs.

**Cause:** Probable discrepancy between the DIS application and the Gateway regarding one or more of the following settings:

- Exercise ID
- Net mask
- Net interface
- IP Address
- UDP Port.

**Resolution:** Ensure the DIS applications and the Gateway are using the same values for Exercise ID, net mask, and UDP port, and that the correct IP address and net interface are specified.

**Problem:** The Gateway is receiving DIS PDUs, but not sending RTI Attribute Updates or Interactions.

**Cause:** The Gateway is not set to the active state.

**Resolution:** Set the Gateway Command for Toggle Interface Mode (see "Gateway Control Commands," on page 5-8). You must activate both the PDU interface and the RTI interface after the Gateway is initialized.

**Problem:** The Gateway is receiving RTI Reflect Attribute or Receive Interactions, but PDUs are not being sent.

**Cause 1:** The Gateway is not set to the active state.

**Resolution:** Set the Gateway Command for Toggle Interface Mode (see "Gateway Control Commands," on page 5-8). You must activate both the PDU interface and the RTI interface after the Gateway is initialized.

**Cause 2:** Required data was not received.

**Resolution:** Some of the PDU translations require specific data before the PDU is sent. See "Mandatory Attributes," on page 1-5 for the appropriate data requirements.

**Problem:** The Gateway hangs up when starting the connection to a federation or during normal execution.

**Cause:** Federates that terminate and do not resign lock up the RTI.

**Resolution:** Remove the orphaned federate using the Federation Execution process.

**Problem:** Weapon fire munition ID fails to map between DIS and HLA.

**Cause:** This problem occurs when the munition object is not known by the Gateway before the WeaponFire or MunitionDetonation is processed. For example, if a missile is fired in DIS and the DIS Fire PDU is received before the missile's Entity State PDU, the Gateway will fail to map the Fire PDU's munition entity ID into the WeaponFire interaction's munition RTI object name.

**Resolution:** Until the missile's Entity State PDU is received and added to the federation the Gateway cannot map between its DIS entity ID and its RTI object name. If possible, modify the DIS application to transmit the missile Entity State PDU before the Fire PDU.

## 4.3. Using the Gateway GUI

The Gateway Graphical User Interface (GUI) provides an interface for the primary Gateway commands. The GUI uses profiles to maintain configurations for connecting to and managing Gateways. A profile contains the information used to connect the GUI to the Gateway and to invoke the Gateway commands.

The Gateway GUI is a Java application that is completely independent of the Gateway application. Use of the GUI is optional. You do not need it to use the Gateway. To run the GUI, you must install a Java Virtual Machine (JVM) for your platform. (The MÄK Gateway runs natively on supported platforms.)

The GUI communicates with the Gateway using point-to-point UDP packets. In order to establish point-to-point communication, the GUI must be given the Gateway's IP address and the GUI-To-Gateway-Port UDP port (defined in the Gateway's configuration file). The Gateway-To-GUI UDP port is used for receiving feedback from the Gateway. A single GUI can communicate with multiple Gateways by switching between profiles.

### 4.3.1. The GUI Display

The initial state of the GUI displays only the command menu for selecting Gateway commands, a drop-down list for selecting profiles, and a Connect button to connect to a Gateway with the selected profile (Figure 4-2).

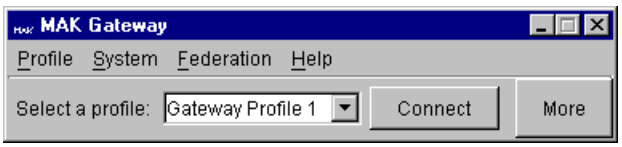



Figure 4-2. Gateway GUI

## GUI Menus

Table 4-1 lists the menu options in the GUI with a brief description of each one.

Table 4-1: Gateway GUI menu commands

| Menu/Command | | Description |
|---|---|---|
| Profile | | Commands for creating and managing profiles. |
| New | | Create a new profile with a default name and no values. The name is "Gateway Profile X", where X is a number representing the number of profiles in the system. |
| Clone | | Creates a new profile that is an exact copy of the current profile.<br>**Note:** Cloning a profile multiple times in a row can cause problems. If the name of the cloned profile gets too long, the profile name box may display incorrectly. Click the more/less button to make the box resize itself. |
| Remove | | Remove the current profile. |
| Save | Current<br>All | Save profile data to the working copy. |
| Reload | Current<br>All | Reload profile data from the working copy. |
| Exit | | Close the GUI. |
| System | | Controls the command line and message displays. It also invokes the HLA and DIS activations. |
| Command Line | | Displays a dialog box for entering text commands to the Gateway. |
| Messages | | Opens a window that displays any messages that the GUI receives from the Gateway. |
| Activations | | Opens a dialog box for activating or deactivating the HLA and DIS interfaces. |
| Federation | | Commands for Federation Management and Declaration Management (see Chapter 5, *Gateway Commands*). |
| Autostart | | Create a federation, join the federation, and publish and subscribe to the object and interaction classes. |
| Create | | Create a federation. |
| Join | | Join the Gateway to a federation. |

Table 4-1: Gateway GUI menu commands

| Menu/Command | | | Description |
|---|---|---|---|
| | Resign | | Resign the Gateway from the federation. |
| | Destroy | | Destroy the federation. |
| | Object Class | Publish<br>Unpublish<br>Subscribe<br>Unsubscribe | [Un]Publish and [un]subscribe object classes. |
| | Interaction Class | Publish<br>Unpublish<br>Subscribe<br>Unsubscribe | [Un]Publish and [un]subscribe interaction classes. |
| | Request Update | Object<br>Class | Request an update of objects or classes. |
| Help | | | |
| | Help | | Displays a text file describing the GUI. |
| | About | | Provides release information. |

## 4.3.2. Connecting to the Gateway

The Gateway GUI is independent of the Gateway itself. To use the GUI, you must connect to the Gateway.

To connect to the Gateway:

1. In the Select A Profile drop-down list, select the profile you want to use to connect to the Gateway.
2. Press the Connect button. The GUI connect to the Gateway using the information in the profile. It also transmits messages containing the toggles for that profile (see "Setting Toggles," on page 4-15.)

### 4.3.3. Managing Profiles

If you press the More button, the Gateway GUI window expands so that you can edit profiles (Figure 4-3). A profile contains values for the parameters used to connect the GUI to the Gateway and to invoke the Gateway commands. The profiles are stored in */bin/data/profiles.xml*. The information in the profile is in eXtensible Markup Language (XML) format.

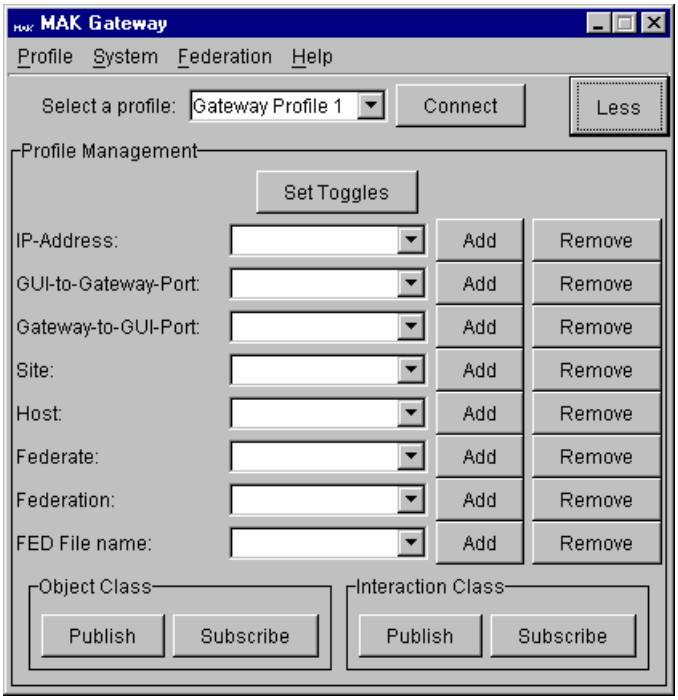



Figure 4-3. Gateway GUI profile management

Each profile option can contain multiple values. A value is active if it is displayed in the text box. Changes to an active value overwrite that value. Blank values, that is, empty or null, are permitted.

Table 4-2 describes each parameter in the Profile Management box.

Table 4-2: Profile options

| Field | Description |
|---|---|
| IP-Address | IP address of the Gateway's computer. |
| GUI-to-Gateway-Port | UDP port for communicating from the GUI to the Gateway. This value can be the same as the Gateway-to-GUI-Port value (see the UDP_PORTS option in "General Configuration," on page 3-3). |

Table 4-2: Profile options

| Field | Description |
|---|---|
| Gateway-to-GUI-Port | UDP port for communicating from the Gateway to the GUI. This value can be the same as the GUI-to-Gateway-Port value (see the UDP_PORTS option in "General Configuration," on page 3-3). |
| Site | Site value used by the Gateway (see "General Configuration," on page 3-3). |
| Host | Host value used by the Gateway (see "General Configuration," on page 3-3). |
| Federate | Federate name used in auto-start, create, and join commands. |
| Federation | Federation name used for auto-start, create, and join commands. |
| Fed-Filename | FED file name used in auto-start, create, and join commands. |

### Opening Profiles

Profiles are loaded when the GUI starts execution and are not saved to the profile file until the GUI is exited. While the GUI is executing, the profile data is maintained in memory as a working copy. Changes made to a profile require an explicit save operation to commit them to the working copy. You can reload a profile from the working copy, thereby eliminating any changes that have not been committed. The commit and reload operations allow changes to be made for temporary or permanent use.

The first time you start the GUI, it opens with a default profile, which is empty.

### Creating A Profile

To create a new profile:

1. In the Gateway GUI window, press the More button to open Profile Management.
2. Choose Profile → New. A default profile name is added to the Select a Profile window. You can edit the name.
3. Specify a value for each of the profile management parameters. Zero (0) is a valid value. (See "Adding Parameter Values to A Profile," on page 4-12.)
4. Set the toggles for this profile. See "Setting Toggles," on page 4-15.
5. Specify object and interaction classes that you want to publish or subscribe. See "Specifying Object Class and Interaction Class Lists," on page 4-14.
6. Choose Profile → Save → Current to save the profile.

### Adding Parameter Values to A Profile

A profile can store multiple values for each parameter. Whichever value is displayed in the Profile Management box is the active value.

To add values to a profile:

1. Click the Add button to the right of the parameter text box. The New *parameter* dialog box opens (Figure 4-4).


Figure 4-4. Add new value dialog box

2. Enter a value for the parameter.
3. Click Done.
4. Choose Profile → Save → Current to save the updated profile.

You can add as many values as you want for each parameter.

> To choose the active value for a parameter, select it from the drop-down list for the parameter.

### Removing A Parameter Value from A Profile

To remove a parameter value:

1. Select the value you want to remove from the drop-down list for the parameter.
2. Click the Remove button for the parameter.
3. Choose Profile → Save → Current to save the updated profile.

### Cloning A Profile

You can copy an existing profile and use it as the basis for creating a new profile.

> To copy a profile, choose Profile → Clone.

A new profile is created. It has the name of the source profile, appended with "(clone)".

### Removing A Profile

To remove a profile:

1. In the Select A Profile drop-down list, select the profile that you want to remove.
2. Choose Profile → Remove.

Once a profile has been removed, the reload operation will not restore it.

### Recovering A Profile

If you inadvertently remove a profile, you can recover it in one of two ways:

- Replace the latest version of *profiles.xml* (the one without the profile you removed) with an older version
- Edit the new version of *profiles.xml.*

To replace the latest version of *profiles.xml*:

1. Copy *profiles.xml* before you exit the GUI.
2. Exit the Gateway GUI. The Gateway GUI saves a new version of *profiles.xml.*
3. Replace the new version of *profiles.xml* with the copy.

---

**!** If you made any changes to profiles besides removing the profile that you want to recover, this method will lose the other changes.

---

To edit the *profiles.xml* file:

1. Copy *profiles.xml* before you exit the GUI.
2. Exit the Gateway GUI. The Gateway GUI saves a new version of *profiles.xml.*
3. In a text editor or XML editor, open both the new and old versions of *profiles.xml.*
4. Copy the profile description for the profile that you deleted from the old version of *profiles.xml* to the new version.

### Specifying Object Class and Interaction Class Lists

At the bottom of the Profile Management box, are two boxes, Object Class and Interaction Class. In each box, there are two buttons, Publish and Subscribe. If you press one of the buttons, a list of the classes used by the Gateway for publication or subscription is displayed. You can edit these lists. There is no practical space limit on these lines (the entire RPR FOM class list is allowed), although their display size is limited.

To edit a publish or subscribe list:

1. Click the Publish or Subscribe button in the Object Class or Interaction Class box.
2. A dialog box opens with an editable text box (Figure 4-5).

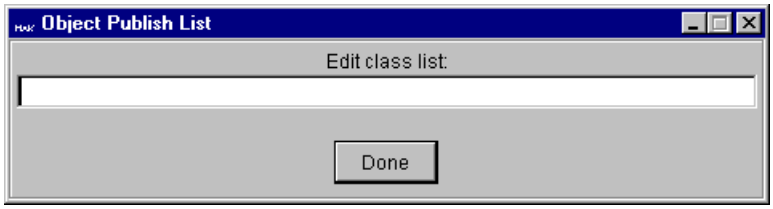



Figure 4-5. Object Publish List dialog box

3. Add objects or interactions by typing their names in the Edit Class List text box.
4. Delete objects or interactions by backspacing over them or by selecting them and pressing delete.
5. Press the Done button.

### Setting Toggles

The Set Toggles button displays the Toggle Settings dialog box (Figure 4-6). This dialog box lets you check or clear toggles for Build Names and Verbose Debug. These check-boxes correspond to toggles within the Gateway.

Build Names determines whether the RTI or the Gateway constructs object names. When checked, the Gateway builds object names using the DIS entity IDs (for example, "198.53.1.ES"). If not checked, the Gateway lets the RTI build object names.

Verbose Debug sets verbose output on (checked) or off (unchecked). This option displays the RTI activities that occur during the Gateway's operation.

To commit changes to the working copy of the current profile and send settings to the Gateway, press Done. Once the changes are committed and saved in the profile, you do not need to execute this dialog box again. The GUI automatically transmits the toggle settings with the first message sent to the Gateway.

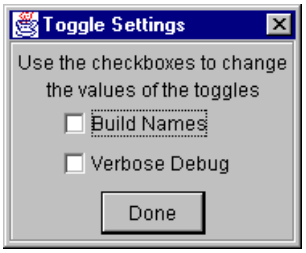



Figure 4-6. Toggle Settings

## 4.3.4. Sending Commands from A Command Line

The GUI does not support all the commands available to the Gateway. If you want to send a command to the Gateway that the GUI does not support:

1. Choose System → Command Line. The Command line dialog box opens (Figure 4-7).

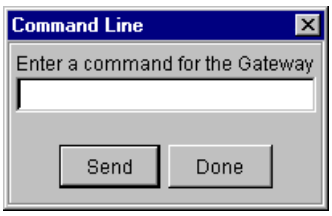



Figure 4-7. Command Line dialog box

2. Enter the text of the command that you want to send.
3. Click Send.

### 4.3.5. Activating and Deactivating the DIS or HLA Interfaces

You can activate or deactivate the HLA and DIS interfaces. This is the equivalent of the `H T` command.

To activate or deactivate HLA or DIS:

1. Choose System → Activations. The Activations dialog box opens (Figure 4-8).

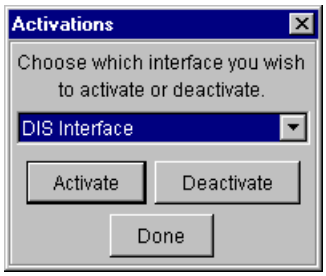



Figure 4-8. Activations dialog box

2. From the drop-down list, select the interface that you want to activate or deactivate.
3. Click the activate or deactivate button.
4. Click the Done button.

### 4.3.6. Displaying Messages from the Gateway to the GUI

From time to time, the Gateway may send messages to the GUI. To display these messages, open the Messages window.

To open the Messages window, choose System → Messages.

# 5

# Gateway Commands

This chapter describes the commands that you can enter to control the Gateway.

# 5.1. Commands

This section describes the various commands used to operate the Gateway. The commands are classified into four categories:

- Federation management
- Declaration management
- Object management
- Gateway control functions.

Table 5-1 summarizes the commands. The commands are described individually in the rest of this chapter.

Table 5-1: Gateway command summary

| Action | Command |
|---|---|
| Federation Management | |
| Create a federation | `H C [federation [FED_file]]` |
| Join a federation | `H J [federate [federation [FED_file]]]` |
| Resign from a federation | `H R` |
| Destroy a federation | `H D Federation` |
| Declaration Management | |
| Publish object classes | `H P[U] O [class]*` |
| Publish interaction classes | `H P[U] I [class]*` |
| Subscribe to object classes | `H S[U] O [class]*` |
| Subscribe to interaction classes | `H S[U] I [class]*` |
| Display Pub/Sub lists | `H L [[P[I\|O] ] \| [S[I\|O] ]]*` |
| Object Management | |
| Request update | `H U Class name` |
| Request update | `H UO Object_handle` |
| Gateway Control Functions | |
| Autostart | `H B [federate [federation [FED_file]]]` |
| Toggle interface active state | `H T [h, p, i, v, o]` |
| Set heartbeat interval, Heartbeat tolerance, or Timeout factor | `H I [es, ee, ld, rr, rt, gw, hb, to,]` |

Table 5-1: Gateway command summary

| Action | Command |
|---|---|
| Protocol Control Functions | |
| PDU text dump | `P D [GPMRITBSLDFCE]*` |
| PDU verbose text dump | `P V [GPMRITBSLDFCE]*`<br>Can list 0 (zero) or more of the following character identifiers. Default: list all identifiers<br>B - Emissions<br>C - Collision<br>D - Detonation<br>E - Entity State<br>F - WeaponFire<br>G - Gridded data (not available)<br>I - IFF<br>L - Logistics (all types)<br>M - Minefield (all, not available)<br>P - Environmental process (not available)<br>R - Radio (Transmitter, Receiver, Signal)<br>S - SIMAN (all types)<br>T - Designator. |
| Echo line to screen | `K E LINE` |
| Execute script file | `K S` *`SCRIPT FILE`* |

## 5.2. Federation Management Commands

Federation management commands let you create, join, and, resign from, and destroy federations. The create and join commands are combined when you use the Autostart command (see "Autostart Gateway," on page 5-8).

### Create Federation

```
H C [Federation [FED-File]]
```

Invokes RTI interface services to create a federation using an optionally specified federation name and FED file. If the names are not specified, the default federation name of VR-Link and *VR-Link.fed* are used. You can specify the default federation and FED file names in the Gateway configuration file (see Chapter 3, *Gateway Configuration*).

The DMSO RTI requires that the FED file exist in the *RTI_HOME/config* directory (or subdirectory).

### Destroy Federation

```
H D [Federation]
```

Invokes RTI interface services to destroy a federation. The default federation is VR-Link. The default federation names can be specified in the Gateway configuration file (see Chapter 3, *Gateway Configuration*).

### Join A Federation

```
H J [Federate [Federation [FED-File]]]
```

Invokes the RTI interface service to join a federation. The default values for federate, federation, and FED file are gateway, VR-Link, and *VR-Link.fed*. You can specify the default federate, federation, and FED file names in the Gateway configuration file (see Chapter 3, *Gateway Configuration*).

### Resign from A Federation

```
H R
```

Invokes the RTI interface service to resign from the federation to which the Gateway is joined. The RTI interface service is invoked using the RTI mechanism to delete objects and release attributes.

## 5.3. Declaration Management Commands

The commands described in this section manage object and interaction publication and subscription. The valid class names that you can use in these commands are sub-strings of the fully qualified FOM class names. The sub-strings must be constructed such that the complete class name is used for each class level, for example, *RadioSignal* or *PhysicalEntity.MilitaryPlatformEntity*. If the Gateway finds a fully qualified FOM class name that contains the string, that FOM class is added to the class names submitted to the RTI.

### Publish Object and Interaction Classes

```
H P[U] O|I [class_list]
```

Invokes RTI Declaration Management services for publication.

| Parameter | Description |
|---|---|
| O | Class list is published as object classes. |
| I | Class list is published as interaction classes. |
| U | Class list is unpublished. |
| class_list | List of classes to publish. |

Default: all subscribable classes of the FOM.

### Subscribe to Object and Interaction Classes

```
H S[U] O|I [list of names to subscribe]
```

Invokes RTI Declaration Management services for subscription.

| Parameter | Description |
|---|---|
| O | Class list is subscribed to as object classes. |
| I | Class list is subscribed to as interaction classes. |
| U | Class list is unsubscribed. |
| class_list | List of classes to subscribe to. |

Default: all subscribable classes of the FOM.

### Display Publication and Subscription Lists

```
H l [[S[O|I]] | [P[O|I]]]*
```

Displays the list of published and subscribed classes. Any combination of options is accepted.

| Parameter | Description |
|---|---|
| `h l` | List all |
| `h l si` | List subscribed interactions |
| `h l so` | List subscribed object classes |
| `h l pi` | List published interactions |
| `h l po` | List published object classes |
| `h l so pi` | List subscribed objects and published interactions |

Default: display all of the lists (`h l`).

## 5.4. Object Management Commands

This section describes commands used for object management. These commands allow the Gateway to join a federation after the initial object data has been sent and get current data for all objects. Both commands are invoked as part of the Autostart command (see "Autostart Gateway," on page 5-8). This ensures that the Gateway receives an invocation of the RTI discoverObject for any existing objects, as well as a complete set of attributes.

### Request Attribute Value Update

```
H U [object_class]
```

This command invokes the RTI requestAttributeValueUpdate service for the object class specified in the command. This command requests a full set of attributes. Class names are matched as described in the introductory paragraph of "Declaration Management Commands," on page 5-5, except that only subscribed classes are considered for matching.

If you do not specify the object class name, the command is invoked using each of the subscribed object classes.

### Request An Attribute Update From A Specific Object

```
H UO rti_object_id
```

This command is a variation on Request Attribute Value Update. It invokes the RTI requestAttributeValueUpdate from the object identified by *rti_object_id*. The full set of attributes is requested.

## 5.5. Gateway Control Commands

This section describes commands for controlling the Gateway.

### Toggle the Interface Mode

```
H T [H,P,V,O]
```

Toggles gateway control modes. When the Gateway first starts up, it does not send any data, so you must set the interface mode.

| Parameter | Description |
|---|---|
| `H` | Controls output to HLA (passes data from DIS to HLA) |
| `P` | Controls output to DIS (passes data from HLA to DIS). |
| `V` | Controls output of verbose text to screen (displays text). |
| `O` | Controls object naming (uses DIS ID names). |

Default: activates DIS to HLA and HLA to DIS output.

With RTI 1.3, all objects are assigned unique string names. The names can be provided automatically by the RTI or the federate can submit names. You can toggle these two methods can be toggled with the `O` command. If object naming is enabled, the DIS IDs are used to create names in dot notation (that is, 17.15.1.ES for entity state objects).

### Autostart Gateway

```
H B [Federate [Federation [FED-File]]
```

This command provides a convenient mechanism for configuring the Gateway in a single command. The Autostart command automatically invokes a sequence of RTI interface commands for Create Federation, Join Federation, Publication, and Subscription. If the initial attempt to join is unsuccessful, the Gateway retries at ten second intervals for a maximum of three attempts. (This is to ensure that the federation executive process has had sufficient time to initialize.) You can repeat the command can be repeated if three attempts are not sufficient. At the conclusion of these events, the Gateway is configured for operation, in an inactive state.

### Set Heartbeat Interval, Timeout Factor and Heartbeat Tolerance

```
H I value list_of_pdu_types | TO | HB
```

Specifies the heartbeat interval for the list of PDU types, the timeout factor (`TO`), and heartbeat tolerance (`HB`).

*`value`* is a floating pointing value.

The list of possible PDU types (specified as two-character names) is:

- `ES` for Entity_State
- `LD` for Designator
- `EE` for Emission
- `RT` for Transmitter
- `RR` for Receiver
- `IF` for IFF
- `GW` for Gateway Update Interval
- `TO` for Timeout Factor
- `HB` for Heartbeat tolerance.

The configuration file, SIM.LOD, also allows you to set these values. Values set by this command override those set in the configuration file (see "Simulation Loading," on page 3-4).

# A

# Configuration Files

This appendix lists the text of Gateway configuration files.



---

**!** Do not change any configuration parameters other than those described in Chapter 3, *Configuring the Gateway*. The parameters that are not described are functionally obsolete, but removing them may cause the Gateway to run incorrectly.

---

# A.1. Sim.cfg

The *sim.cfg* file configures network information.

```
*
*   @(#) Module: GATEWAY/sim/ sim.cfg
*   @(#) Version: 2.4    Last modified: 24 Sep 1997
*
************************************************************************
*                                                                      *
*  Simulator Configuration Data                                        *
*                                                                      *
*  Warning, values for the following must be enclosed in quotes:       *
*       NET_MASK, NET_INTERFACE, IP_ADDRESS, UDP_PORTS, TDB, UTM_COORDS,    *
*       SCRIPT_FILE_NAME, EW_RDB, EW_EDB, EW_PDB, EW_CFG                *
*                                                                      *
************************************************************************/

*******************************
*  Communications information  *
*******************************
site =     1       * REQUIRED: IST is site 17
host =          1             * REQUIRED: host is machine ID number
exercise =      1               * REQUIRED: exercise ID of this simulation

ip_address = "0.0.0.0"   * IP addresses for machine
net_mask   = "0.0.0.0"    * Broadcast mask for the network

*net_interface = "ec2"* Specify alternative network interface for
                        * simulation traffic (DIS/SIMNET) when more
                        * than one network card is used.

udp_ports = "3000,3001,3002,3003"* UDP port numbers for DIS and Internal
       Messages
                       * (DIS, OUTPUT, MSG_SEND, MSG_RECEIVE)

udp_check =    OFF          * UDP checksum mode:
                            *   OFF      - no checksum computations
                            *   GENERATE - send checksums with packets
                            *   VALIDATE - verify incoming checksums
                            *   ON       - both GENERATE and VALIDATE

*********************************
*  Terrain Database information  *
*********************************
* tdb = "TEST.MCC"* REQUIRED: filepath of terrain database
* tdb = "GRAF.MCC"* REQUIRED: filepath of terrain database
* tdb = "HL_1994.MCC"* REQUIRED: filepath of terrain database
* tdb = "KNOX.MCC"* REQUIRED: filepath of terrain database
* tdb = "HLPC.MCC"* REQUIRED: filepath of terrain database

ext_coord_system = geocentric   * For DIS only!  SIMNET will use local
                                * coordinates.
                                * Should geocentric or local coordinates
                                * be used?  If geocentric is specified
                                * then UTM coordinates must be choosen.
                                * geocentric - Default option
                                * local      - local coordinates used in PDUs
                               * If LOCAL is choosen then the program is non-
                                * DIS compliant!
```

```
* UTM coordinates of terrain database: center x, center y, zone
* (required to convert Simulator coordinates to DIS protocol):

        * utm_coords =  "669982.00, 5452418.000, 32"
        * Grafenfels SIMNET terrain database
        * The UTM Coordinates for the GRAF database are questionable.  The
        * values above are maintained for reference, should questions arise,
        *  but the following coordinates are assumed to be correct

        *utm_coords = "596000.000, 3431000.000, 14"
        * Fort Knox SIMNET terrain database (both "tinyknox" in release and
        Knox
        * TDB available from LORAL).

        *utm_coords =   "634996.683842,3953199.527991,10"
        * 50km x 50km SIMNET terrain database.  (Hunter TDB available from
        LORAL)
        * That is, latitude N 35 deg 42 min 48.63 sec,

        *utm_coords = "585000.0,3903000.0,10"
        * 100km x 100km Pre-SIF I/ITSEC terrain database.

        *utm_coords = "649184.676,3972381.199,10"
        * HDA and terrain database.

        *utm_coords = "584909.625,3901166.849"
        * Full SIF TDB

        utm_coords = "650000.000,3975000.000,10"
        * SIF 10x30 high resolution area, I/ITSEC '93

        gw_cfg = "gateway.cfg"

*********************************
*  EW Database information       *
*********************************
*EW_RDB = "e:\simdb\receiver.dat"
*EW_PDB = "e:\simdb\parms.dat"
*EW_EDB = "e:\simdb\emit.dat"
*EW_CFG = "sim.ew"

*********************************
*  Miscellaneous information     *
*  Optional                      *
*********************************
logo                    = no    * Display logo?
*stealth_host           = 23    * host of a standard SIMNET stealth
*create_from_remote     = yes   * Allow remote entity creation?
*random_seed            = 0     * If missing or zero seed is set from the
        system
                                * clock, otherwise given value is the seed.
*script_file_path       = "."   * Path for script files

*** END OF FILE ***
```

## A.2. Sim.lod

The *sim.lod* file configures the interface.

```
*
*       @(#) Module: GATEWAY/sim/ sim.lod
*       @(#) Version: 2.1Last modified: 4/15/97
*
*****************************************************************************
*                                                                           *
*  SAFDI Simulator Load Management Data                                     *
*                                                                           *
*  Each option has a comment describing its purpose.  Each option is either *
*  "REQUIRED" or has a "DEFAULT" value listed (which would be used if the   *
*  option were commented out or didn't appear).                             *
*                                                                           *
*  Most of the optional configuration settings have been set to their       *
*  default values.                                                          *
*                                                                           *
*  An asterisk (*) indicates a comment.                                     *
*                                                                           *
*****************************************************************************

************************
*  Entity limitations  *
************************
max_internal_entities  = 64      * Max number of internal entities that can be
                                 *   created.  DEFAULT is 12, max is 64
max_sightable_entities = 40      * Max number of entities that will be
                                 *   considered for LOS calculations.
                                 *   DEFAULT is 40

************************
* Set patch buffer    *
************************
num_patch_buffs = 1000            * X is the number of patch buffers that can
                                 * be used.

************************
*  Interface options  *
************************
text =          yes              * Display text? (printfs and DisplayMessages)
keyboard =      yes              * Allow keyboard input?
*graphics =     yes               * Display graphics?
input_window = yes
graphics =      no               * Display graphics?
*input_window = no

*****************************************************************************
*  Intervals for updating periodic behaviors of entities in 1/100ths of *
*  a second (i.e. 100 = 1 second).                                          *
*****************************************************************************
dynamics =      20               * Dynamics update interval.  DEFAULT is 20
* dead_reckon =  20                 * Dead Reckoning update interval.  DEFAULT
        is 20
los =           50               * LOS update interval. DEFAULT is 500
emission =      500              * EW update interval. DEFAULT is 500

statistics =    yes              * Gather statistics? (-g or -p will override
        NO)
```

```
************************
*  Performance options  *
************************
machine_speed = 66.0            * Machine speed in megahertz.  DEFAULT is 50.

                                * Simulation costs for...
simulation_costs = (0.1         *   ..Static_Mgr
                    0.1         *   ..Dynamics_Mgr
                    1.0         *   ..Static_Entity
                    8.0)        *   ..Dynamic_Entity
                                * Values are unitless and are used for ratios
                                * only.  DEFAULT is (0.1,0.1,1.0,8.0)

********************************
* Network communication options *
********************************

*****************************************************************************
*  Activate filter on the specified PDU types. The PDU will be filtered     *
*  at the application protocol level.                                       *
*  Choices are as follows:                                                  *
*                                                                           *
*   appearance              deactivate              impact                  *
*   fire                    indirect_fire           collision               *
*                                                                           *
*   emission                laser_designator                                *
*                                                                           *
*   service_request         resupply_offer          resupply_received       *
*   resupply_cancel         repair_complete         repair_response         *
*                                                                           *
*   create_entity           remove_entity           acknowledge             *
*   start_resume            stop_freeze             set_data                *
*   data                    data_query              action_request          *
*   action_response         message                 event_report            *
*   stealth_appearance                                                      *
*****************************************************************************

*filter_pdu =
*(
*appearance
*)

send_network_packets = yes
receive_network_packets = yes

*** END OF FILE ***
```

## A.3. Gateway.cfg

The *gateway.cfg* file configures HLA federation information.

```
*****************************************************************************
*  Gateway Interface Configuration Data
*
*  Warning:
*
*  Max. line lenght is 120, including comments and white space
*****************************************************************************

Federate_Name= "Gateway"
Federation_Name= "VR-Link"
Fed_File   = "VR-Link.fed"

Verbose_Output= no
Send_Internal= no
Build_ObjectNames= no

*****************************************************************************
*  Note:
*    The given strings are matched against the fully qualified FOM class
*    names with exact matching performed for each individual class name.
*    The name inside the parenthesis must be quoted.
*    The names must be complete class names
*
*****************************************************************************

*SUBSCRIBE_INTERACTIONS = ( "Detonation" "WeaponFire" "Collision" )
*PUBLISH_INTERACTIONS= ( "Detonation" "WeaponFire" "Collision" )

*SUBSCRIBE_OBJECTS= ( "BaseEntity" )
*SUBSCRIBE_OBJECTS= (
       "BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.Mili
       taryLandPlatform" )
*PUBLISH_OBJECTS= ( "BaseEntity" )

*****************************************************************************
*       Note:
*           Assignment identifiers are fully qualified object class names
       derived
*           from BaseEntity:!
*****************************************************************************
*ENTITY_TYPE_MAPPING =
*(
*
       BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.Milit
       aryAirLandPlatform = ( (1 1 225 1 1 0 0) )
*
       BaseEntity.PhysicalEntity.MilitaryEntity.MilitaryPlatformEntity.Milit
       aryAirLandPlatform = ( (1 1 225 1 1 0 0) )
*)

*****************************************************************************
```

# B

# References

This appendix lists reference material that was used to help develop the Gateway and which you may find of interest in expanding your understanding of DIS, HLA, the RPR FOM, and interoperability issues.


Cox, A. and Petty, M. D., (1998), "Use of DIS Simulation Management (SIMAN) in HLA Federations," *Proceedings of the 1998 Spring Simulation Interoperability Workshop*, Orlando FL, March 9-13, 1998.

Cox, A., Petty, M. D., and Wood, D. D. (1997), "RTI 1.0 Experiments and Results," *Proceedings of the 1997 Fall Simulation Interoperability Workshop*, Orlando FL, September 8-12, 1997.

Cox, A., Wood D. D., Petty, M. D., and Juge, K. A. (1996a). "Integrating DIS and SIMNET Simulations into HLA with a Gateway," *Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Orlando FL, September 16-20 1996, pp. 517-525.

Cox, A., and Wood, D. D. (1996b), "Design and Implementation of the BDS-D/HLA Gateway," *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, Orlando FL, December 3-6 1996.

Harkrider, S. M. and Petty, M. D. (1996a), "Results of the HLA Platform Proto-Federation Experiment," *Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Orlando FL, September 16-20 1996, pp. 441-450.

Harkrider, S. M. and Petty, M. D. (1996b), "High Level Architecture and the Platform Proto-Federation," *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, Orlando FL, December 3-6 1996.

IEEE (1995), Standard for Distributed Interactive Simulation—Application Protocols, IEEE Standards Department, Piscataway, NJ, 1995

Pope, A. R., The SIMNET Network and Protocols, (1991) Version 6.6.1, BBN

SISO-RPR (1999a), Real-Time Platform Reference FOM, Simulation Interoperability Standards Organization Reflector - Plug and Play Family of Models, http://chat.sc.ist.ucf.edu/confs/SISO-RPR, 1997

SISO-RPR (1999b), Guidance Rational Interoperability Modalities for the Real-Time Platform Reference FOM, Simulation Interoperability Standards Organization RPR FOM SDG, http://www.sisostds.org/stdsdev/rpr-fom/index.htm, 1999

Wood D. D. (1999). "A One-To-Many / Many-To-One Transformation in an HLA Gateway Between the DIS Electromagnetic Emissions PDU and the RPR FOM," *Proceedings of the 1999 Fall Simulation Interoperability Workshop*, Orlando FL, September 1999.

Wood D. D. (1998). "DIS Radio Protocol Representation and Translation in the RPR FOM," *Proceedings of the 1998 Spring Simulation Interoperability Workshop*, Orlando FL, March 9-13, 1998.

Wood, D. D., Petty, M. D., Cox, A., Hofer, R., and Harkrider, S. M. (1997a). "HLA Gateway Status and Future Plans," *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, Orlando FL, March 3-7, 1997, pp. 807-814.

Wood D. D. (1997b). "An Object-Oriented RTI Interface," *Proceedings of the 1997 Fall Simulation Interoperability Workshop*, Orlando FL, September 8-12, 1997.

Wood, D. D., Cox, A., and Petty, M. D., (1997c), "An HLA Gateway for DIS Applications," *Proceedings of the 19th Interservice/Industry Training Systems and Education Conference*, Orlando FL, December 1-4 1997.

# Index

## D



## E



## F

## M



## N



## O



## P

## R



## S

## T



## U



## V



## W


***MÄK Technologies***
*185 Alewife Brook Parkway*
*Cambridge, MA 02138*
*617.876.8085*

GWY-3.4-1-000612