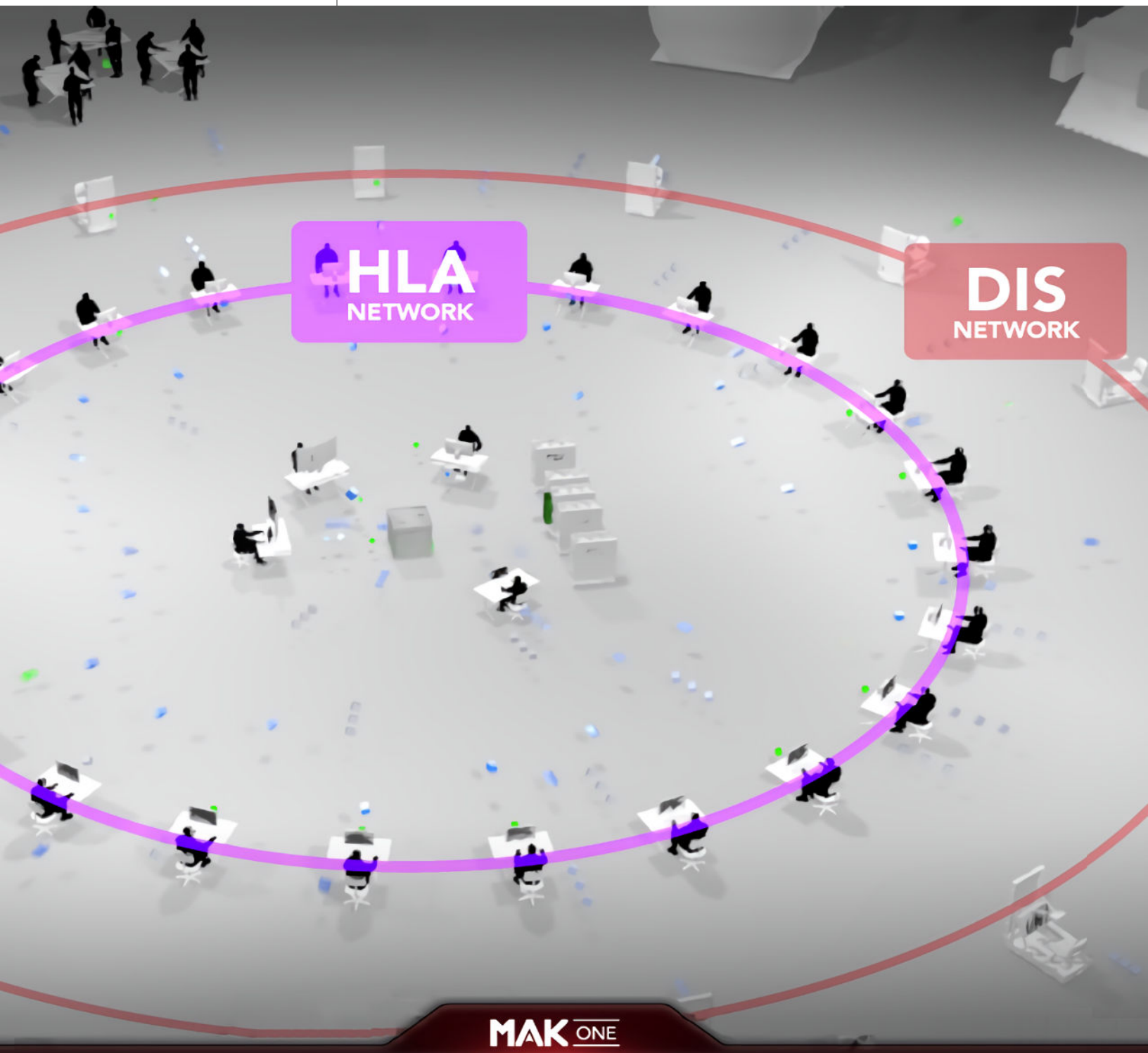




Copyright © 2025 MAK Technologies
All rights Reserved. Printed in the United States.

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

VR-Exchange™, VR-TheWorld™, VR-Vantage™, DI-Guy™, and DI-Guy Scenario™ are trademarks of MAK Technologies. MAK Technologies®, VR-Forces®, RTIsPy®, B-HAVE®, and VR-Link® are registered trademarks of MAK Technologies.

GL Studio® is a registered trademark of The DiSTI® Corporation.

Portions of this software utilize SpeedTree® technology (©2008 Interactive Data Visualization, Inc.). SpeedTree is a registered trademark of Interactive Data Visualization, Inc. All rights reserved.

RFView® is a registered trademark of Information Systems Laboratories (ISL), Inc.

SilverLining™ is a trademark of Sundog Software.

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

3ds Max® is a registered trademark of Autodesk, Inc.

All other trademarks are owned by their respective companies.

For third-party license information, visit the MAK ONE Product Support Knowledge Base (at <https://www.mak.com/knowledge>) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).

MAK Technologies
10 Fawcett Street, Suite 204
Cambridge, MA 02138 USA

Voice: +1 617.876.8085

Fax: +1 617.876.9208

info@mak.com

www.mak.com

Revision RTI-5.0-08-250702



Contents

Preface

How the Manual is Organized	12
Documentation Set	13
Try the Help	13
Links Between PDFs	13
MAK Products	14
How to Contact Us	16
Telephone	16
E-mail	16
Internet	16
Post	16
Document Conventions	16
Mouse Button Naming Conventions	17
Third-Party Licenses	17

1. Introduction to the MAK RTI

1.1. About RTIs	20
1.1.1 Obtaining HLA RTI Specification Documents	20
1.2. Compatibility with Other RTI Implementations	20
1.2.1 Dynamic Linking	21
1.2.2 The FED File and FDD File	21
1.2.3 The RTI Initialization Data File	22
1.2.4 Network Compatibility	22
1.3. The RTIs Spy Plug-in API	22
1.4. Support for the HLA 1516 Specification	22
1.4.1 RTI Interoperability	23

2. MAK RTI Concepts

2.1. Using the MAK RTI with Multithreaded Applications	26
2.2. The MAK RTI Process Model	26
2.2.1 The Role of the tick() Function in the MAK RTI	27
2.2.2 Asynchronous I/O	28
2.2.3 Asynchronous Message Processing	28
2.2.4 Asynchronous Callbacks	29
2.3. Network Topologies for RTI Messages	29
2.3.1 Best Effort Transport on a LAN	29
2.3.2 Reliable Transport and TCP Forwarding on a LAN	29
2.3.3 Best Effort Transport on a WAN	30
2.3.4 Centralized TCP Forwarding on a WAN	30
2.3.5 Distributed Forwarding on a WAN	30
2.4. Federate Protocol	30
2.4.1 Federate Protocol Use Cases	31
2.4.2 C++ Federate Protocol Wrapper	31

3. Compiling and Linking with the MAK RTI

3.1. Library Names	34
3.2. Compiling and Linking with HLA 1.3	35
3.2.1 Compiling and Linking on Linux	35
3.2.2 Compiling and Linking on Windows	35
3.3. Compiling and Linking with HLA 1516	36
3.3.1 Compiling and Linking on Linux	36
3.3.2 Compiling and Linking on Windows	36
3.4. Compiling and Linking with HLA Evolved	37
3.5. Compiling and Linking with HLA 4	37
3.5.1 Compiling and Linking on Linux	37
3.5.2 Compiling and Linking on Windows	38
3.6. Federation Time	38
3.6.1 The HLA 1.3 Time Implementation	38
3.6.2 The RTI 1516 Time Implementation	39
3.6.3 The HLA Evolved Time Implementation	39
3.6.4 The HLA 4 Time Implementation	40
3.7. Java Bindings for the MAK RTI	40

4. RTI Settings

4.1. The RTI Initialization Data (RID) File	42
4.2. Selecting the RTI Settings to Use	42
4.2.1 Search Order for RID Files	42
4.3. Editing RTI Settings	43

4.3.1 Adding Custom Configurations	57
4.3.2 Importing RTI Settings	58
4.3.3 Deleting RTI Settings Configurations	58
4.4. Displaying Configuration Settings in the Federations View	59
4.5. Configuring HLA Switch Values	60
4.6. Configuring Use of Advisory Messages	60
4.7. Consistency of RTI Settings	60
4.8. Specifying RTI Settings Programmatically in HLA 1516	61
4.9. The RTI Settings File Consistency Checker	62
4.9.1 Enabling RTI Settings Consistency Checking	62
4.9.2 Selecting the Parameters to be Checked	63

5. RTI Applications

5.1. The rtiexec	66
5.1.1 Configuring the rtiexec	66
5.1.2 Configuring Diagnostic Messages	67
5.2. The RTI Assistant	67
5.2.1 Specifying the RTI Assistant's Log Update Interval	68
5.2.2 Shutting Down the RTI Assistant	69
5.2.3 Configuring the Display of RTI Assistant Messages	69
5.2.4 Connecting to Remote Assistants	69
5.2.5 Specifying the Warning Tick Interval	71
5.2.6 Common Operations in RTI Assistant Log Windows	71
5.2.7 Specifying the RTI Assistant's Ports	72
5.2.8 Specifying the RTI Assistant's Multicast Address	73
5.2.9 Specifying the RTI Assistant's Multicast Time-to-Live	73
5.2.10 Disabling the RTI Assistant	73
5.2.11 RTI Assistant Command-Line Options	73
5.3. The RTI Forwarder Application	74
5.3.1 Starting an RTI Forwarder	74
5.3.2 Shutting Down an RTI Forwarder	77
5.4. The RTI Federate Protocol Server	77
5.4.1 Starting the RTI Federate Protocol Server	78
5.5. The Federation Shadow Application	78
5.5.1 RTI Federation Shadow Command-Line Options	79
5.6. The RTI Command Line Tool	80
5.6.1 Listing Federation Executions and Federates	82
5.6.2 Deleting Federates	82

6. Managing Federations

6.1. Introduction	84
6.2. Specifying the FED File	84

6.2.1 Search Order for FED Files	84
6.2.2 Distributing the FED File	84
6.3. Using Lightweight Mode	85
6.3.1 Limitations of Lightweight Mode	86
6.4. Viewing Information about a Federation	87
6.4.1 The Network Map	88
6.5. Viewing Time Management State Information	88
6.6. Viewing Synchronization Points	89
6.7. Viewing Federation-Wide Notification Messages	90
6.8. Configuring Fault Tolerance Strategies	91
6.8.1 Responding to Abnormal Termination	91
6.8.2 Reconnecting to the RTI Forwarder	94
6.8.3 Specifying the Response Interval	95
6.8.4 Processing Discovery of Unknown Objects	96
6.8.5 Choosing Silent Failure Instead of Exceptions	96
6.9. Configuring Save/Restore	97

7. Managing Federates

7.1. Displaying Federate Information	100
7.2. Displaying FOM Object and Interaction Classes	100
7.3. Viewing LRC Diagnostic Messages	101
7.3.1 Enabling LRC Diagnostic Data Logging	102
7.3.2 Displaying the Federate Log	102
7.4. Configuring Federate Connections in the RTI Settings	103

8. Monitoring the Network

8.1. Introduction	106
8.2. Testing Latency Between Federates	106
8.2.1 Enabling Latency Testing	107
8.2.2 Running Latency Tests	107
8.2.3 Interpreting the Results of Latency Tests	108
8.2.4 Running Latency Tests Manually	110
8.2.5 Running Continuous Latency Tests	111
8.2.6 Configuring Latency Tests	111
8.3. The RTIsPy Network Monitoring Tool	112
8.3.1 Enabling Federate Statistics Monitoring	112
8.3.2 Displaying Object Statistics	113
8.3.3 Displaying Interaction Statistics	114
8.3.4 Displaying RTI Message Statistics	115
8.3.5 Displaying RTI Services Statistics	115
8.3.6 Displaying Callback Statistics	116
8.3.7 Graphing Federate Statistics	116

8.4. Examples of Using the RTIsPy Network Monitoring Tool	118
8.4.1 Finding Bottlenecks Caused by Processing Callbacks	118
8.4.2 Identifying the Cause of Slowdowns Under Time Management	118
8.4.3 Comparing the Costs and Benefits of RTI Features	119

9. Managing Network Traffic

9.1. Choosing Best Effort or Reliable Transport	122
9.2. Using the Correct IP Address Format	123
9.3. Packet Bundling for Reliable and Best Effort Messages	123
9.4. Automatically Locating the RTI Forwarder and rtiexec	124
9.5. Sender-Side Filtering (Smart TCP Forwarding)	124
9.5.1 Configuring Sender-Side Filtering	126
9.5.2 Configuring Sender-Side Filtering for rtiexec Messages	127
9.6. Compressing RTI Messages	127
9.7. Enabling Support for Large Attributes and Parameters	128
9.8. Configuring the Network Buffer Sizes	129
9.9. Controlling Congestion for Best Effort Sockets	130
9.10. Configuring the Processing Model	130
9.11. Buffering Automatic Update Requests (1516 Specifications Only)	131
9.12. Filtering by Multicast Groups Using Declaration Management	132
9.12.1 Assigning DM Multicast Groups to Network Interfaces	133

10. Configuring Message Forwarding

10.1. Configuring UDP and Centralized TCP Forwarding on a LAN	136
10.1.1 Configuring Best Effort (UDP) on a LAN	136
10.1.2 Configuring Centralized TCP Forwarding on a LAN	136
10.2. Configuring UDP and Centralized TCP on a WAN	137
10.2.1 Configuring Centralized TCP Forwarding	137
10.2.2 Configuring UDP Over a WAN Without Distributed Forwarding	138
10.3. Distributed Forwarding Topologies on a LAN	139
10.3.1 Forwarder Groups	139
10.3.2 Singleton Forwarders	140
10.3.3 Advanced Distributed Forwarding on a LAN	141
10.4. Distributed Forwarding Topologies on a WAN	142
10.4.1 Optimizing Distributed Forwarding across a WAN	143
10.5. Configuring Federates for Distributed Forwarding	146
10.6. Configuring RTI Forwarders for Distributed Forwarding	146
10.6.1 Configuring RTI Forwarders Using the RTI Assistant	146
10.6.2 Creating a Routes File	147
10.6.3 UDP Forwarding	149
10.6.4 Configuring Single-Portal Forwarder Groups	150
10.6.5 Configuring Load-Balanced Forwarder Groups	151

10.6.6 Configuring Distributed UDP Forwarding	152
10.7. Connecting to Remote Forwarders using the RTI Assistant	154
10.7.1 Connecting to Remote Forwarders on the Command Line	155
10.8. Configuring Routers or Firewalls for WAN Federations	155
10.9. Special Distributed Forwarder Networks	155
10.9.1 Configuring Hierarchical Forwarder Networks	155

11. Configuring HLA Services

11.1. Enabling and Disabling HLA Services	160
11.1.1 Configuring MOM Services	160
11.1.2 Configuring Time Management Services	162
11.2. Configuring Internal Messages	163
11.2.1 Configuring the Delay Time for Processing Internal Messages	163
11.2.2 Transmitting Internal Messages Across Ticks	164

12. Using Data Distribution Management

12.1. Configuring DDM Services	168
12.1.1 Enabling DDM	168
12.1.2 Configuring Distributed Region DDM	169
12.1.3 Configuring a Fixed Grid DDM	169
12.1.4 Configuring Implicit DDM	171
12.2. The Distributed Region Approach	173
12.2.1 Attribute Passelization	174
12.3. The Fixed Grid Approach	175
12.3.1 The Fixed Grid Representation	176
12.3.2 Federate Region Intersections with the Fixed Grid	177
12.3.3 Normalization of Federate Data for Range Bounds	180
12.3.4 The Fixed Grid Approach and DDM Services	180
12.4. Implicit DDM	182
12.4.1 Implicit DDM Implementation	183
12.4.2 Adding Implicit DDM to the FED or FDD	183
12.4.3 Implicitly Associating DDM Regions	184
12.4.4 Modifying Region Extents	185
12.4.5 Modifying Fixed Grid Region Extents	186

13. Using FOM Modules

13.1. Using FOM Modules	188
13.2. Merging FOM Modules into the Current FOM	188
13.2.1 Using FOM Modules in Lightweight Mode	189
13.3. Creating and Joining with FOM Modules	190
13.4. Validating FOM Modules	190

13.5. Configuring Use of Modular FOMs	191
13.5.1 Configuring a MOM Module	192
13.5.2 Configuring Create FOM Modules	192
13.5.3 Configuring Join FOM Modules	192
13.5.4 Configuring Use of a Local FOM Module File	193
14. Update Rate Reduction	
14.1. Update Rate Reduction	196
14.2. How Update Reduction Is Implemented	197
14.3. Configuring Update Rate Reduction	198
14.3.1 Specifying the Update Rate in 1.3 and 1516	199
14.3.2 Specifying the Rate for Object Class Subscriptions in 1.3 and 1516	200
14.3.3 Specifying the Receive Filtering Tolerance	200
14.3.4 Example Update Rate Reduction Configuration (1.3 and 1516 Only)	200
A. RTI Settings Reference	
A.1. MÄK Technologies Lisp (MTL) Files	204
A.1.1 Using Environment Variables in an MTL File	205
A.1.2 Using Conditional Statements	205
A.1.3 Converting Between Strings and Numbers	205
A.2. RTI Settings	205
A.3. Federation-Wide Parameters	206
A.4. LRC-Specific Parameters	217
A.5. Configuring tick(min,max)	233
B. Example Federates	
B.1. Example Federates	236
B.2. Building the Examples	236
B.3. The rtisimple Example	236
B.4. The simpleDDM Example	237
B.5. The simpletime Example	238
B.6. HLA Bounce	240
B.6.1 Running HLA Bounce	240
B.6.2 Demonstrating Ownership Management	241
B.6.3 Demonstrating DDM	242
B.7. The MAK Shooter Example	242
B.7.1 Building MAK Shooter	243
B.7.2 Running MAK Shooter	243
B.7.3 Playing MAK Shooter	244
Index	247



Preface

This manual is for developers of HLA simulation applications and for persons who need to install or use the MAK RTI.

This manual assumes that you are familiar with the command-line and graphical windowing environments for your operating system.

For information about changes to the MAK RTI since this manual went to press, please see MAK RTI Release Notes.

How the Manual is Organized

This manual is organized as follows:

"Chapter 1. Introduction to the MAK RTI" on page 19 describes the MAK RTI and its features.

"Chapter 2. MAK RTI Concepts" on page 25, provides conceptual information about the MAK RTI that will help you understand when and why you would want to use particular features and configuration options.

"Chapter 3. Compiling and Linking with the MAK RTI" on page 33, explains how to link applications to the MAK RTI.

"Chapter 4. RTI Settings" on page 41 explains how to maintain and distribute RTI initialization data (RTI settings) files. It does not describe parameters. Parameters are described in "Appendix A. RTI Settings Reference" on page 203.

"Chapter 5. RTI Applications" on page 65 describes the rtiexec, the RTI command line tool, the RTI Forwarder, and how to configure the RTI Assistant.

"Chapter 6. Managing Federations" on page 83 explains how to use the RTI Assistant to manage federations. It also describes lightweight mode and fault tolerance strategies.

"Chapter 7. Managing Federates" on page 99 explains how to use the RTI Assistant to manage federates.

"Chapter 8. Monitoring the Network" on page 105 describes how to use the RTI Assistant and RTIspy GUI to monitor the performance of federates. It also describes how to run latency tests.

"Chapter 9. Managing Network Traffic" on page 121 describes parameters used to configure the RTI's interaction with the network, except for those used to configure message forwarding.

"Chapter 10. Configuring Message Forwarding" on page 135 explains basic message forwarding strategies, the different types of distributed forwarding topologies, and how to configure them.

"Chapter 11. Configuring HLA Services" on page 159 explains how to configure MOM and time management services and how to configure internal messages.

"Chapter 12. Using Data Distribution Management" on page 167 provides conceptual background for the MAK RTI's DDM implementations.

"Chapter 13. Using FOM Modules" on page 187 explains the MAK RTI's implementation of FOM modules.

"Chapter 14. Update Rate Reduction" on page 195 explains the MAK RTI's implementation of update rate reduction.

"Appendix A. RTI Settings Reference" on page 203 describes the format of the RTI settings file and each parameter.

"Appendix B. Example Federates" on page 235 describes the sample federates provided with the MAK RTI.

Documentation Set

MAK RTI documentation is provided as manuals in PDF format, online help, and class documentation. Electronic versions of the documentation are in `./makRtix.x/doc`. On Windows, you can access the documentation from the MAK RTI folder on the Start menu. You can also find the most current documentation at <https://www.mak.com/mak-one/support/help/specs/mak-rti>. The MAK RTI documentation set includes:

- The *MAK RTI User Guide* is a brief introduction to HLA and the MAK RTI and explains how to configure the MAK RTI startup environment and run a federate.
- The *MAK RTI Reference Manual* provides detailed information about configuring the MAK RTI and designing network topologies for HLA federations.
- MAK RTIs API class documentation and MAK RTI API developer documentation in HTML format. You can access the class documentation from the Start menu in Windows or by opening `./classdoc/index.html`.
- HTML5 help. The MAK RTI has HTML help, which you can access from the RTI Assistant system tray icon's context menu, the Federations View window, the Documentation shortcuts folder, or `./doc/help`. It has a natural language search and combines both the *MAK RTI User Guide* and the *MAK RTI Reference Manual*.
- The *MAK ONE Products Interoperability Guide* explains how MAK products work together and provides troubleshooting information.
- The *MAK RTI Release Notes* provides system information, new features and updates, a list of fixed bugs, and a list of known problems.

Try the Help

MAK's HTML5 help is the best way to use our documentation. It's easy to navigate, with improved natural language search tools and filters to focus your search. You can click the illustrations to see high-resolution enlargements that show more detail. It's built into the product and installed locally — simply press **F1** in the Federations View or choose Help from the RTI Assistant context menu to launch the help in your browser. You do not need to be connected to the internet to use it.

 If you attempt to view the help in a browser that does not fully support HTML5, you may experience issues with some features.

Links Between PDFs

Many MAK PDF documents link to each other. For example, the description of a new feature in the release notes might include links to detailed information in a user guide, and there are also many links from user guides to the *Adding Content to MAK ONE Applications Reference Manual* and back. These links between PDFs rely on two things:

- The linked PDF files must be in the same directory. MAK installers put the PDFs for each product in that product's `doc` directory — in Windows, `./makRti/doc`. If you move or copy a PDF away from that directory or download a new one to another

location, any links between that PDF and the others stop working.

- You must use Acrobat Reader to read MAK PDFs. Links between PDFs do not work when you read with a web browser. Acrobat Reader is free; you can get it from Adobe at <https://get.adobe.com/reader>.

MAK Products

The MAK Technologies product line includes:

- **VR-Link® Network Toolkit.** VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the NETN FOM and RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications. VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.
- **MAK RTI.** An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MAK RTI is optimized for high performance. It has an API, RTIsPy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- **VR-Forces®.** VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you 2D and 3D views of a simulated environment.

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

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

- **VR-Vantage™.** VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes VR-Vantage and the VR-Vantage Toolkit.
 - VR-Vantage displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to simulation objects so that it moves as they do.

- SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics-based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
- The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™. VR-TheWorld is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- RFView® for MAK ONE. Synthetic Aperture Radar (SAR) imaging is an indispensable tool for today's modern militaries. Developed in partnership with ISL, RFView for MAK ONE acts as a SAR server and delivers simulated SAR images that are both physically accurate and fully correlated with out-the-window views and EO/IR images of MAK ONE synthetic environments. It includes a toolkit that enables you to build a client application to request SAR images from the server.
- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several

sample applications that work with VR-Forces and VR-Vantage.

- MAK Legion. MAK Legion is a collection of applications and an SDK designed to support simulations with millions of simulation objects.

How to Contact Us

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

Telephone

Call or fax us at:	Voice:	617-876-8085
	Fax:	617-876-9208

E-mail

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

Internet

MAK website home page:	www.mak.com
License key requests:	https://www.mak.com/mak-one/support/portal
Technical support:	https://www.mak.com/mak-one/support/portal
Product version and platform information:	https://www.mak.com/mak-one-specs
For the free, unlicensed MAK RTI:	https://www.mak.com/mak-one/support/help/bonus-material

Post

Send postal correspondence to:	MAK Technologies 10 Fawcett Street, Suite 204 Cambridge, MA 02138 USA
--------------------------------	---

When requesting support, tell us the product you are using, the version, and the platform on which you are running. If you are using VR-Vantage, VR-Forces, VR-Engage, or DI-Guy, please also include your data compatibility version and date.

Document Conventions

This manual uses these typographic conventions:

Sans Serif Bold	Indicates a key on the keyboard or menu options. Mentions of letter keys do not require you to press the Shift key. If Shift is necessary, that will be included, for example, Shift+Q .
Monospaced	Also indicates parameters.
<i>Monospaced</i>	Indicates values you enter such as commands or arguments.
<i>Italic</i>	Indicates command variables that you replace with appropriate values.

{ }	Indicates required arguments.
[]	Indicates optional arguments.
	Separates options in a command where only one option may be chosen at a time.
()	In command syntax, indicates equivalent alternatives for a command-line option, for example, (<code>--help -h</code>).
/	Indicates a directory. Since MAK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames.
<i>Italic</i>	Indicates a file name, pathname, or a class name.
►	Indicates a procedure.
Menu > Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose File > Save .
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the MAK RTI home directory. For example, the directory `./makRti5.0/doc` appears in the text as `./doc`.

Mouse Button Naming Conventions

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

Third-Party Licenses

MAK software products may use code from third parties. For an up-to-date list of supported third-party and open source libraries and the associated license information, please visit the MAK ONE Product Support Knowledge Base (at <https://www.mak.com/knowledge>) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).



1. Introduction to the MAK RTI

An RTI, such as the MAK RTI, is an implementation of the Run-Time Infrastructure (RTI) portion of the High-Level Architecture (HLA).

1.1. About RTIs	20
1.1.1 Obtaining HLA RTI Specification Documents	20
1.2. Compatibility with Other RTI Implementations	20
1.2.1 Dynamic Linking	21
1.2.2 The FED File and FDD File	21
1.2.3 The RTI Initialization Data File	22
1.2.4 Network Compatibility	22
1.3. The RTIsPy Plug-in API	22
1.4. Support for the HLA 1516 Specification	22
1.4.1 RTI Interoperability	23

1.1. About RTIs

The MAK RTI is a software library (and supporting executables) that implements the High Level Architecture (HLA) 1.3, HLA 1516 (SISO DLC HLA API 1516), HLA Evolved (IEEE 1516-2010), and HLA 4 (IEEE 1516-2025) interface specifications. The HLA was developed by the U. S. Department of Defense (DoD) to provide a means for the many different types of simulations in use by the DoD to interoperate. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications must use an RTI. (Applications that use VR-Link typically do not need to make RTI calls directly, but only because VR-Link makes the RTI calls for them.) The MAK RTI has been optimized for the high performance needs of the real-time simulation community.

The MAK RTI has been verified by the Modeling and Simulation Coordination Office (MSCO, formerly DMSO) as fully compliant with the HLA Interface Specification, version 1.3 and with the IEEE 1516 version of the HLA Interface Specification (SISO DLC HLA API 1516 (SISO-STD-004.1-2004)). All services are implemented.

The US DoD RTI Verification activity was suspended due to budget constraints in January 2015 before declaring any HLA Evolved (IEEE 1516.1-2010) RTIs officially verified. However, at the time that the RTI verification activity was suspended, the MAK RTI had successfully passed 100% of the available tests.

No verification has been announced for HLA 4 (IEEE HLA 1516-2025). To learn about it, read "[HLA Evolves Again](https://www.mak.com/learn/blog)" on the mak.com blog (<https://www.mak.com/learn/blog>).

 When we use the terms "HLA 1516" or "1516", we mean the SISO DLC HLA API 1516 specification. When we use the term "HLA Evolved", we mean IEEE 1516-2010. When we use the term "HLA 4", we mean IEEE 1516-2025. When we discuss a subject that applies equally to HLA 1516, HLA Evolved, and HLA 4, we refer generically to the "HLA 1516 specifications".

1.1.1 Obtaining HLA RTI Specification Documents

The HLA 1.3 and all IEEE 1516 specifications are copyrighted by their owners and MAK cannot distribute them to customers. You can find documentation for the 1516 interface specifications at:

- <https://ieee.org>
- <https://sisostds.org>

1.2. Compatibility with Other RTI Implementations

This section discusses aspects of compatibility:

- Dynamic linking
- Static linking
- The FED file and FDD file
- The RTI Initialization Data (RID) file (also called the RTI settings file)
- Network compatibility

1.2.1 Dynamic Linking

The MAK RTI 1.3 implements the API dictated by the 1.3 version of the HLA Interface Specification, as amended by MSCO's official HLA 1.3 Interpretations document. All RTIs that are verified as compliant with the HLA 1.3 specification are written to this exact API standard. This ensures that the MAK RTI is link-compatible with other RTIs that have been verified as HLA compliant.

The MAK RTI 1516 implements the SISO DLC HLA API¹ 1516 (SISO-STD-004.1-2004). This API supports dynamic link compatibility, which was problematic with the original IEEE 1516 API. The IEEE 1516 API only supports compile-time compatibility (it uses C++ templates and implementation-specific header files). The MAK RTI is compatible with any RTI written to the SISO DLC HLA API 1516.

The MAK RTI HLA Evolved implements the IEEE 1516-2010 standard. HLA Evolved incorporates the SISO Dynamic Link Compatible API. Therefore, federates compiled with the HLA Evolved version of MAK RTI can be used with other RTIs that support the specification without being recompiled. Other important features include Modular FOMs, update rate reduction, improvements to reliability and connectivity, and other changes that make using HLA easier.

Because an RTI is typically accessed by an application through a dynamic library (*.dll* on Windows, *.so* on Linux), applications do not need to be recompiled or relinked to switch among dynamic-link-compatible RTI implementations that have been built using the same compilers. The only requirement is that the appropriate version of the library be located within the shared library search path. For more information, see "Chapter 3. Compiling and Linking with the MAK RTI" on page 33.

1.2.2 The FED File and FDD File

In both the HLA 1.3 specification and the 1516 specifications, you must provide FOM data to the RTI. In the 1.3 specification, this is done with the federation execution data (FED) file. The 1516 specifications use a FOM Document Data (FDD) file. However the format is slightly different between 1516 and HLA Evolved.

All versions of the MAK RTI support both the FDD and FED file formats. FDD files must have a file extension of either *.xml* or *.fdd*. FED files are expected to have an extension of *.fed* or *.mtl*; however, any other extensions are assumed to be in the FED format.

The FED file format is part of the HLA 1.3 standard, so the MAK RTI 1.3 uses the same FED file format as other compliant RTIs. You do not need to modify your FOM or FED file to switch between the MAK RTI 1.3 and other RTI 1.3 implementations.

There may be differences among RTI implementations as to where the FED file must be located. By default, the MAK RTI requires that the FED file must be available to be read by each federate in the federation. An alternative is for the federation execution to distribute the FED file to each Local RTI Component (LRC) when the federate joins. For more details, see "6.2. Specifying the FED File" on page 84.

¹Simulation Interoperability Standards Organization Dynamic Link Compatible High Level Architecture Application Program Interface

i Since the FED file and FDD file perform the same function, and since for the MAK RTI they are essentially interchangeable, in MAK RTI documentation, unless otherwise noted, references to the FED file should be considered as references to the FED file and the FDD file.

Before you run an HLA application, you must make sure that your federates (and the rtiexec if you are running it) can find the FED file for the federation you are running. For details, see "6.2. Specifying the FED File" on page 84.

1.2.3 The RTI Initialization Data File

The RTI initialization data (RID) file (or RTI settings file) is the configuration file for an RTI. The MAK RTI RID file is written using MAK Technologies Lisp (MTL). Unless you want to rely on default values for all RTI configuration parameters, you need to make sure that all federates (and the rtiexec if you are running it) can find the appropriate RID files. For details, see "Chapter 4. RTI Settings" on page 41.

1.2.4 Network Compatibility

The MAK RTI is not network compatible with RTI implementations from other vendors. This means that within a given federation, you cannot run one federate with the MAK RTI and another federate with a different RTI. In general, no two RTI implementations are network compatible, due to differences in the low-level network mechanisms that they use (which include, but are not limited to packet layout).

The term Run-Time Infrastructure refers to the entire system used to communicate among federates. By HLA's design, choosing an RTI is a federation-wide choice, and not a federate-by-federate choice. Applications that want to interoperate in the same federation must all use the same RTI. A federation can easily switch from one RTI implementation to another between runs, but during each run, all participants must agree on which RTI to use, much as they must also agree on which FOM to use.

1.3. The RTIs Spy Plug-in API

The RTIs Spy Plug-in API allows you to write code that can monitor the internal state of the RTI and modify the RTI's behavior. For information about the RTIs Spy Plug-in API and how to load plug-ins, please see the HTML API documentation.

1.4. Support for the HLA 1516 Specification

The MAK RTI 1516 supports the full set of services defined by the HLA 1516 interface specification. The MAK RTI 1516 implements SISO-STD-004.1-2004.

The C++ API that was published as part of the IEEE 1516 standard had two significant problems. First, because it was published before any reference implementations had been developed, it contained a series of bugs that made it impossible to implement as-is. Second, its heavy reliance on templates made it impossible to assure that different RTI implementations would be link-compatible.

The SISO Dynamic Link Compatibility HLA API Product Development Group (HLA API PDG, <https://www.sisostds.org>) developed an alternative mapping of the IEEE 1516 Interface Specification to a C++ API (SISO-STD-004.1-2004). This alternative 1516 API addresses the bugs in the API that was published along with the IEEE standard, and also allows for dynamic-link compatibility among different RTI implementations. As part of this process, the HLA API PDG migrated the API away from the use of templates for defining the API classes and towards simpler C++ declarations.

Some of the key changes are:

- The use of the "pImpl" idiom for handles (decouples the class interfaces from their private member declarations)
- The use of simple enumerations (which are strongly typed in C++)
- Passing parameters by reference instead of `auto_ptr` (except for factories).

Although the Dynamic Link Compatible 1516 API more closely resembles the 1.3 API than the original 1516 API does, some significant differences remain. For example, an RTI namespace is used instead of a top-level RTI class, templates are still used to represent handles (to allow different implementations of handles), and the API uses Standard Template Library (STL) maps to represent sets of attribute handle/value pairs.

By implementing against the SISO 1516 API, MAK promotes link compatibility with other RTI vendors, which is not possible with the original IEEE 1516 API. The IEEE 1516 API only allowed compile-time compatibility (especially true with the use of templates). However, link compatibility is important because it allows federates built using one RTI to execute with the local RTI component (LRC) library and central RTI component (CRC) executable of a different RTI without having to recompile the federate.

i While the 1516 API uses the `std::wstring` to exchange string information, internally, the MAK RTI 1516 operates on narrow strings only (that is, representable by `std::string`). Therefore, the contents of any wide string passed through the RTI API must be convertible to a narrow string.

1.4.1 RTI Interoperability

All versions of the MAK RTI use the same code base. As a result, they share the same on-the-wire protocol for exchanging internal RTI information. For a subset of RTI services, a federate using the MAK RTI HLA Evolved can interoperate with a federate using the MAK RTI HLA 4. However, the differences in the functionality or format between the HLA Evolved and HLA 4 APIs create some conflicts that do not allow interoperability:

- Ownership management is not interoperable between the HLA Evolved and HLA 4 APIs. HLA 4 adds a user-supplied tag argument to several of the ownership management service calls and callbacks. When using ownership management services between HLA Evolved and HLA 4 federates, the user-supplied tag arguments may be dropped or left unfilled in certain conditions.
- The MOM loaded is dependent on the federate that creates the federation. When an HLA Evolved federate creates a federation, the HLA Evolved MOM is used. Likewise, when an HLA 4 federate creates a federation, the HLA 4 MOM is used. There are

several changes between the HLA 4 MOM and the HLA Evolved version. There are additional federate attributes for all switches, and new federation attributes for every federation switch. There is a new HLA 4 feature: service reporting to a file, as well as changes to the HLAreportServiceInvocation Interaction. The availability of these MOM features is dependent on the version of HLA used by the federate that creates the federation. This means startup order of the federates can change the MOM services that are available.

- FOM modules are interoperable between HLA Evolved and HLA 4.
- Directed Interactions are not supported in HLA Evolved. HLA Evolved federates will not receive them, even as non-directed interactions.
- In HLA Evolved, there is an HLAconveyProducingFederate switch that can be enabled or disabled. In HLA 4, that switch no longer exists in the FOM and defaults to always enabled. In mixed federations, this switch is forced to be "True" for all HLA Evolved federates, as well.
- DDM is interoperable between HLA Evolved and HLA 4, but the way dimensions are declared has changed between standards. In HLA Evolved, dimensions were specified per attribute while in HLA 4 they are specified on the object class level.
- Update Rate Reduction works across protocols, as long as the RTI settings file defines the same update rates as are present in the HLA Evolved FOM and HLA 4 FOM.

 HLA 4 is new and therefore not all potential interoperability issues were known at the time this document was published. Please visit the MAK ONE Product Support Knowledge Base (<https://www.mak.com/knowledge>) to look for issues discovered after the release.



2. MAK RTI Concepts

This chapter provides a brief introduction to how MAK has implemented the RTI specifications. This information will help you understand and put to use the various configuration options described in later chapters.

2.1. Using the MAK RTI with Multithreaded Applications	26
2.2. The MAK RTI Process Model	26
2.2.1 The Role of the tick() Function in the MAK RTI	27
2.2.2 Asynchronous I/O	28
2.2.3 Asynchronous Message Processing	28
2.2.4 Asynchronous Callbacks	29
2.3. Network Topologies for RTI Messages	29
2.3.1 Best Effort Transport on a LAN	29
2.3.2 Reliable Transport and TCP Forwarding on a LAN	29
2.3.3 Best Effort Transport on a WAN	30
2.3.4 Centralized TCP Forwarding on a WAN	30
2.3.5 Distributed Forwarding on a WAN	30
2.4. Federate Protocol	30
2.4.1 Federate Protocol Use Cases	31
2.4.2 C++ Federate Protocol Wrapper	31

2.1. Using the MAK RTI with Multithreaded Applications

The MAK RTI LRC is thread-safe, but by default, it is not re-entrant. The LRC is only re-entrant when using the asynchronous callback process model described in "2.2.4 Asynchronous Callbacks" on page 29.

The concept of thread-safety versus re-entrancy is very important for multi-threaded federates. Since the MAK RTI LRC is thread-safe, multiple federate threads can create RTI and Federate Ambassadors within a single process. This is important for several classes of federate such as bridge federates, which may join multiple federations simultaneously. So, thread-safety means that it is safe to use single or multiple instances of the LRC in a multi-threaded federate, but it does not mean that multiple federate threads can access a single RTI Ambassador instance.

To optimize performance for the majority of federates, the MAK RTI is not re-entrant by default. To guarantee re-entrancy, the RTI would have to lock internally during each access. Locking of this sort at the API level can create a significant amount of overhead, which offers no benefit in most federates. This means that all access to a single RTI Ambassador instance must be made by a single federate thread. If multiple threads require access to the LRC, then the federate is responsible for ensuring correct locking of all data passed to the RTI. In general, the federate is able to perform more selective and optimal locking than the RTI through its knowledge of interactions between the federate threads. Therefore, the recommended practice is to create a separate thread for RTI access and provide thread-safe communication from the RTI thread to the other federate threads.

If a federate requires multi-threaded access to a single LRC instance and cannot follow the procedure described above, then the federate must use the asynchronous callback processing model. When asynchronous callbacks are enabled, multiple federate threads can access the RTI Ambassador simultaneously, and the RTI will provide Federate Ambassador callbacks in a new thread created by the RTI. However, the asynchronous callback model usually increases federate complexity significantly.

2.2. The MAK RTI Process Model

The MAK RTI supports these process models:

- Fully synchronous. All RTI processing is done in one thread and RTI network I/O, message processing, and federate ambassador callbacks are evoked when the federate calls `tick()`.
- Asynchronous networking/synchronous message processing and callbacks. Network input is handled in a separate thread, but the input is buffered, not processed. Message processing and callbacks are handled in the main thread when the federate calls `tick()`.
- Asynchronous networking and message processing/synchronous callbacks. Network input and message processing is handled in a separate thread. Callbacks are cached and delivered in the main thread when the federate calls `tick()`.
- Fully asynchronous. Network I/O, message processing, and callbacks are handled in a separate thread. The callbacks are delivered asynchronously and these calls

provided by the federate code must be thread safe. The `tick()` call becomes a NULL operation and the federate does not need to call it.

For details about configuring the process model, see "9.10. Configuring the Processing Model" on page 130.

2.2.1 The Role of the `tick()` Function in the MAK RTI

In the synchronous, asynchronous I/O, and asynchronous messages processing models, it is important to understand the role of the `tick()` function in HLA 1.3 and the `evokeCallback()` and `evokeMultipleCallbacks()` functions in HLA 1516 and HLA Evolved. For simplicity, the term tick will be used unless a distinction is necessary. Proper use of `tick()` is critical to federate and federation performance. As the names of the `evokeCallback()` and `evokeMultipleCallbacks()` member functions suggest, the primary purpose of `tick()` is to give the RTI processing cycles that it can use to invoke callbacks through the Federate Ambassador. In both the synchronous and asynchronous I/O models, Federate Ambassador callbacks are delivered only during the processing of the `tick()` function. For example, updates and interactions are delivered only when the federate calls `tick()`. Similarly, time advances can only be granted during the `tick()` call. For this reason alone, all federates (excluding federates using the asynchronous callbacks processing model) must call `tick()` regularly and at appropriate times (for example, after a name reservation or subscription) to ensure smooth operation within the federation.

Except for the asynchronous message processing and asynchronous callback modes, almost all internal RTI processing is performed during the `tick()` call. This provides federates with much greater control of the RTI's CPU usage, but it also means that federates must tick often in order to prevent starving the RTI of CPU time. This can be a major problem for any RTI algorithm that requires multiple messages to pass between joined federates or between federates and the `rtiexec`. Algorithms that require multiple message exchanges progress only during the call to `tick()`, and may require multiple calls to `tick()`. For example, the time management algorithms employed by the MAK RTI sometimes require multiple messages between a federate and the `rtiexec` for a single Time Advance Grant. This means that if a federate does not tick frequently while waiting for the grant, then the time advance will slow down accordingly – this can affect the performance of all time constrained federates.

Synchronous mode federates are even more sensitive to tick frequency than federates using asynchronous I/O. Federates operating in synchronous mode read data from the network only during `tick()`. Therefore, ticking too slowly may lead to two distinct problems. First, if the federate ticks infrequently, then its local network buffers may overflow, leading to dropped best effort messages. Secondly, once the federate's network buffer overflows, reliable messages will begin to build up at the federate's RTI Forwarder. The forwarder will then gradually increase its memory usage until the federate clears the backlog of messages.

Finally, federates occasionally connect to the RTI (by creating an RTI Ambassador) well before they intend to participate in a federation. Such federates must still tick the RTI (even before they have joined) in order to avoid the reliable messaging problem mentioned previously. This problem affects all synchronous and asynchronous I/O mode federates.

2.2.2 Asynchronous I/O

Asynchronous I/O allows a federate to better schedule its time and to better respond to peaks in I/O processing.

Many of the RTI services require data to be transmitted over the network. The federate LRC creates a separate I/O thread that processes network I/O operations separately from other internal RTI processing. The asynchronous I/O thread separates the network transmission portion of this processing from the federate's thread of execution. Since the network I/O is not performed during the processing of RTI Ambassador service calls, the RTI returns the thread of execution to the federate significantly faster. The federate is better able to respond to time-critical processing that might in turn generate an increased output load (for example, aircraft maneuvering). The I/O thread can also use spare CPU cycles when the federate is waiting for other system operations (for example, disk access). Similarly, the asynchronous I/O thread receives data from the network separately from the federate's thread of execution.

The I/O thread only performs network I/O operations. The federate must still invoke the `tick()` member function to allocate CPU cycles to the RTI for its internal processing and to generate federate ambassador callbacks. However, the federate is less likely to affect data transmission (for example, dropped messages or blocked TCP connections) if it fails to invoke the `tick()` function frequently enough (for example, during initialization phases).

Disadvantages of Asynchronous I/O

A disadvantage of asynchronous I/O is that it can increase message transmission latency. The timing of when messages are sent and received becomes less determined, since the federate LRC thread is no longer performing the actual I/O operations. However, the MAK RTI uses a signaling strategy between the threads that allows the latency to be small while the RTI is not heavily loaded, and for it to take advantage of queuing when loads increase. The threads signal each other when messages are available for sending and receiving.

Another disadvantage of asynchronous I/O is that the federate thread can fail to yield enough processing cycles to the I/O thread. If the I/O thread becomes starved (that is, it cannot get enough processing cycles), it will not process network I/O. The RTI messages will not be sent or received, resulting in high latency and buffer overflows. In addition, operating system mechanisms can take effect that can be detrimental to federate performance. (For example, Windows gives starved threads control for some number of cycles.)

You can configure the interaction between the federate LRC thread and the I/O thread with parameters in the RTI settings (Table 12 on page 206). One of these parameters controls whether the LRC yields to the I/O thread during the `tick()` call (on by default). Having the federate LRC thread yield during the `tick()` call ensures that the I/O thread will be executed. We recommend that this parameter value should not be changed unless the federate has been designed to manage multithreading issues so that it sufficiently yields execution to other threads.

2.2.3 Asynchronous Message Processing

Asynchronous message processing is a process model that is halfway between asynchronous I/O, which simply supports asynchronous message buffering, and asynchronous callbacks, which supports fully asynchronous processing and callbacks. In the asynchronous message processing mode, the asynchronous thread handles the

network I/O and processes received messages. However, any federate ambassador callbacks that are generated enter a queue to be delivered to the federate synchronously during its call to `tick()` or the `evokeCallback` RTI Ambassador service.

This processing mode relieves the federate from having to call `tick()` to provide the RTI with processing cycles. It may improve the response and efficiency of some distributed algorithms (for example, time management). However, it requires the caching of callback parameters, which results in additional memory allocation and data copy operations.

2.2.4 Asynchronous Callbacks

The MAK RTI supports an asynchronous callback mode. Instead of buffering messages to be processed during the next RTI `tick()`, the asynchronous thread processes the messages directly. In this mode, the federate does not have to invoke `tick()`, because all internal RTI processing is performed in the RTI Ambassador calls or the asynchronous thread. This configuration requires that the Federate Ambassador callbacks be thread-safe.

Asynchronous callbacks are enabled in the RTI settings.

 Specifying a callback model of `HLA_IMMEDIATE` in the `RTIambassador` connect service in HLA Evolved forces the RTI into asynchronous callbacks mode, regardless of the RTI settings.

2.3. Network Topologies for RTI Messages

The RTI can send messages using best effort transport (UDP) or reliable transport (TCP). It provides alternatives for optimizing performance both within a LAN and over a WAN.

2.3.1 Best Effort Transport on a LAN

When you use best effort transport within a LAN, messages are sent to the other federates on the LAN via UDP multicast or broadcast. There is no guarantee of receipt. Configuration is simple and there is little overhead. To configure best effort transport, configure the MAK RTI as described in "9.1. Choosing Best Effort or Reliable Transport" on page 122.

2.3.2 Reliable Transport and TCP Forwarding on a LAN

Reliable transport guarantees receipt of messages. When you send data to multiple recipients using TCP, a separate copy of each packet must be sent to each recipient. In general, there are two approaches for achieving this:

- A sender can maintain a TCP connection to each remote federate and can send a copy directly to each.
- A sender can maintain a connection to one or more RTI Forwarders – applications that receive TCP traffic from a sender, and then forward it to many recipients.

The first approach minimizes latency in federations that have a small number of federates, because packets reach their destinations in a single network hop. However, the second approach minimizes CPU processing time at each federate and minimizes the number of connections.

The MAK RTI uses TCP forwarding for all reliable transmissions. It supports centralized TCP forwarding (one forwarder) using the RTI Forwarder application. The MAK RTI supports distributed TCP forwarding through the use of multiple RTI Forwarder applications. This allows you to distribute the total amount of network traffic among the forwarders. In large federations, this can improve performance over that of a single centralized forwarder. For details about how to configure reliable transport on a LAN, see "10.1. Configuring UDP and Centralized TCP Forwarding on a LAN" on page 136.

2.3.3 Best Effort Transport on a WAN

Most routers do not support broadcast or multicast messages. Therefore, to send best effort messages over a WAN, you must specifically configure the addresses of the recipients or use a packet forwarding strategy. For details of distributed UDP forwarding, please see Distributed Forwarding later in this section. For details about configuring UDP over a WAN without packet forwarding, see "10.2.2 Configuring UDP Over a WAN Without Distributed Forwarding" on page 138.

2.3.4 Centralized TCP Forwarding on a WAN

The centralized TCP forwarding configuration for the MAK RTI works over a WAN. However it provides no additional performance benefits. For WAN communications, distributed forwarding is the recommended strategy.

2.3.5 Distributed Forwarding on a WAN

The MAK RTI's distributed forwarding option supports both TCP and UDP. The distributed forwarder allows federation designers to create a network of forwarders, each of which manages a subset of the federates in the federation. Distributed forwarding has two main benefits:

- It spreads the network load across multiple forwarders, rather than putting it all on a single forwarder.
- It achieves optimal network bandwidth use by reducing the number of message copies sent across the WAN from one per federate to one copy, or even none (if you are using sender-side filtering).

To configure distributed forwarding, see "Chapter 10. Configuring Message Forwarding" on page 135.

2.4. Federate Protocol

The previous sections of this chapter apply to federates written using the C++ and Java APIs of the HLA standard. The HLA 4 version of the HLA standard introduces a new message-based API named the Federate Protocol.

The Federate Protocol API defines a set of messages to be exchanged between a federate and the RTI over the network using a point-to-point network connection based on TCP/IP. Once a session is established between a federate and the RTI, interactions with the RTI are carried out by sending and receiving network messages. A federate invokes a service call by sending a service call message to the RTI which returns a service call response message in acknowledgement and to provide any return arguments if necessary. Likewise, the RTI will send a service callback message to the federate which the federate uses to

carry out Federate Ambassador callback invocations. Upon receiving a service callback message from the RTI, the federate returns a service callback response message in acknowledgement. Further details about the Federate Protocol can be found in the IEEE 1516-2025 Interface Specification.

2.4.1 Federate Protocol Use Cases

The Federate Protocol message-based API opens new use cases for how HLA applications connect and integrate with federations. By implementing RTI services and service calls through messages, any software capable of opening a TCP connection can theoretically implement the Federate Protocol and participate in an HLA exercise, where previously only the C++ and Java APIs existed. This facilitates other programming languages, such as Python, C#, Javascript, or others to connect to an RTI.

Additionally, the point-to-point network connection between the Federate Protocol Server and the Federate Protocol client allows for simple connectivity over a WAN, making cloud-based federates and federations easier to connect. With only one pipeline of data between the client and the server, monitoring and securing the connection simplifies security requirements.

2.4.2 C++ Federate Protocol Wrapper

The MAK RTI provides a C++ wrapper around the Federate Protocol that presents the same API as the HLA RTI's C++ API. All management of establishing and terminating a Federate Protocol session is handled within the C++ API connect and disconnect service calls. The Federate Protocol session uses port 15164 by default, but you can change it using the `RTI_FEDPRO_PORT` environment variable. Maintaining the connection via sending heartbeat messages is handled during the `evoke(Multiple)Callack(s)` call. Therefore, the federate must make service calls (which serve as a heartbeat) or call `evoke(Multiple)Callack(s)` with an interval not to exceed the heartbeat interval, which is 60 seconds by default. A federate should consider the session connection broken if it does not receive a service call response, service callback message, or heartbeat response from the RTI within the heartbeat timeout period (180 seconds by default). The heartbeat interval and heartbeat timeout values are currently not configurable via the RID.

An RTI service call is implemented as a blocking operation that sends a service call request message and waits for the service call response. The federate provides an implementation of the Federate Ambassador, which the wrapper uses to make callbacks. Currently, the Federate Protocol Wrapper only supports an `EvokeCallback` callback model. During the `evoke(Multiple)Callback(s)` invocation, the wrapper receives service callback request messages from the RTI and invokes the appropriate Federate Ambassador callback and sends the service callback response messages back to the RTI.

The current implementation of the federate protocol wrapper currently only supports the basic set of service calls and service callbacks listed below. The remaining service calls result in the RTI throwing an internal error exception.

Service Calls:

- connect (with configuration)
- disconnect
- list federations execution

- list federations execution members
- create federation execution (with FOM modules and time implementation name)
- destroy federation execution
- join federation execution
- resign federation execution
- publish object class attributes
- subscribe object class attributes
- publish interaction class
- subscribe interaction class
- reserve object instance name
- register object instance (class handle and name)
- update attribute values (object handle, attribute values, user supplied tag)
- delete object instance (object handle, user supplied tag)
- send interaction (class handle, parameter values, user supplied tag)

Service callbacks:

- connection lost
- report federation executions
- report federation execution members
- report federation execution does not exist
- federate resigned
- object instance name reservation succeeded
- object instance name reservation failed
- discover object instance
- remove object instance
- reflect attribute values (object handle, attribute values, user supplied tag, transport, federate handle, region handle pointer (null))
- receive interaction (class handle, parameter values, user supplied tag, transport, federate handle, region handle pointer (null))

 The create federation execution and join federation execution FOM module parameters currently expect simple filenames. The use of URLs and FOM content is not yet supported.

At this time, the MAK Federate Protocol Wrapper API uses a different library name than the HLA C++ API. As a result, a federate built with the Federate Protocol Wrapper is not link compatible with a federate built with the HLA C++ API. However, they both use the same header files and the same federate source code can be compiled and linked using either of the API's libraries.

An example Federate Protocol federate is in the *./examples* directory.



3. Compiling and Linking with the MAK RTI

This chapter provides details about how to compile and link applications with the MAK RTI.

3.1. Library Names	34
3.2. Compiling and Linking with HLA 1.3	35
3.2.1 Compiling and Linking on Linux	35
3.2.2 Compiling and Linking on Windows	35
3.3. Compiling and Linking with HLA 1516	36
3.3.1 Compiling and Linking on Linux	36
3.3.2 Compiling and Linking on Windows	36
3.4. Compiling and Linking with HLA Evolved	37
3.5. Compiling and Linking with HLA 4	37
3.5.1 Compiling and Linking on Linux	37
3.5.2 Compiling and Linking on Windows	38
3.6. Federation Time	38
3.6.1 The HLA 1.3 Time Implementation	38
3.6.2 The RTI 1516 Time Implementation	39
3.6.3 The HLA Evolved Time Implementation	39
3.6.4 The HLA 4 Time Implementation	40
3.7. Java Bindings for the MAK RTI	40

3.1. Library Names

The procedures for compiling and linking applications with the MAK RTI are essentially the same for each version of the RTI. The main difference is that they have different library names. The MAK RTI 64-bit architecture for all platforms. For HLA 1.3 and HLA 1516, the 64-bit libraries append `_64` to the base library names. For HLA Evolved, they use the prescribed terminology (e64).

If you try to run a 64-bit HLA 1.3 or 1516 application that was compiled with an RTI that does not use the `_64` convention, that application will find the 32-bit libraries instead of the 64-bit libraries and will not run. You can work around this problem by deleting the 32-bit libraries and creating a symlink between the 32-bit library name and the 64-bit library name. Alternatively, you can recompile the application using the MAK RTI libraries.

For HLA 4, there is a SISO Reference standard (SISO-REF-075-2023) for library name schemas. Library names now include a compiler marker token to specify the compatible compiler. For Windows, the compiler marker value is `vc141`. For Linux distributions, the compiler marker is `gcc8` for Red Hat 8 and `gcc11` for Red Hat 9.

Table 1. MAK RTI libraries

HLA Version	Platform	Compiler	Library Names
HLA 1.3	Windows	Visual Studio 2017 64-bit	<i>libRTI-NG_64.lib</i> and <i>.dll</i> <i>libfedtime_64.lib</i> and <i>.dll</i>
	Red Hat 8	GCC 8 64-bit	<i>libRTI-NG.so</i> <i>libfedtime.so</i>
	Red Hat 9	GCC 11 64-bit	<i>libRTI-NG.so</i> <i>libfedtime.so</i>
HLA 1516	Windows	Visual Studio 2017 64-bit	<i>librti1516_64.lib</i> and <i>.dll</i> <i>libfedtime1516_64.lib</i> and <i>.dll</i>
	Red Hat 8	GCC 8 64-bit	<i>librti1516_64.so</i> <i>libfedtime1516_64.so</i>
	Red Hat 9	GCC 11 64-bit	<i>librti1516_64.so</i> <i>libfedtime1516_64.so</i>
HLA Evolved	Windows	Visual Studio 2017 64-bit	<i>librti1516e64.lib</i> and <i>.dll</i> <i>libfedtime1516e64.lib</i> and <i>.dll</i>
	Red Hat 8	GCC 8 64-bit	<i>librti1516e64.so</i> <i>libfedtime1516e64.so</i>
	Red Hat 9	GCC 11 64-bit	<i>librti1516e64.so</i> <i>libfedtime1516e64.so</i>

Table 1. MAK RTI libraries (continued)

HLA Version	Platform	Compiler	Library Names
HLA 4	Windows	Visual Studio 2017 64-bit	<i>librti1516_2025vc141.lib</i> and <i>.dll</i> <i>libfedtime1516_2025vc141.lib</i> and <i>.dll</i> <i>libauthorizer1516_2025vc141.lib</i> and <i>.dll</i>
	Red Hat 8	GCC 8 64-bit	<i>librti1516_2025gcc8.so</i> <i>libfedtime1516_2025gcc8.so</i> <i>libauthorizer1516_2025gcc8.so</i>
	Red Hat 9	GCC 11 64-bit	<i>librti1516_2025gcc11.so</i> <i>libfedtime1516_2025gcc11.so</i> <i>libauthorizer1516_2025gcc11.so</i>

3.2. Compiling and Linking with HLA 1.3

► To build an application that uses the MAK RTI HLA 1.3:

1. Include *RTI.hh* in any source files that use RTI functionality.
2. Add the *./include/HLA13* and *./lib* directories to the appropriate search paths.
3. Link with the RTI libraries.

3.2.1 Compiling and Linking on Linux

If you have installed the MAK RTI in, for example, */usr/products/makRti5.0*, this should display on your compile line:

```
-I/usr/products/makRti5.0/include/HLA13
```

and this should display on your link line:

```
-L/usr/products/makRti5.0/lib -lRTI-NG -lfedtime
```

3.2.2 Compiling and Linking on Windows

Assume that the MAK RTI is installed in *C:\MAK\makRti5.0*. (The Microsoft Visual C++ user interfaces vary slightly between versions, so the following description is somewhat genericized.)

► To compile for HLA 1.3, on the C/C++ section of the Project Property Pages:

1. In the General section, add *C:\MAK\makRti5.0\include\HLA13* to the Additional Include Directories property.
2. In the Preprocessor section, add *RTI_USES_STD_FSTREAM=1* to the Preprocessor Definitions (ostream is in the std namespace).
3. For MSVC++ 8 or 9, set *_SECURE_SCL* to 0. For VC++ 10, use the default value.
4. For MSVC++ 8 or 9, set *_HAS_ITERATOR_DEBUGGING* to 0. For VC++ 10, use the default value.

5. For MAK RTI plug-ins, add `C:\MAK\makRti5.0\include\MAK` to the Additional Include Directories property.
6. In the Code Generation section, in the Run-time Library property, select Multithreaded DLL (/MD) for release builds. Select Multithreaded Debug DLL (/MDd) for debug builds.

► **To link to the HLA 1.3 libraries, on the Linker section of the Project Property Pages dialog box:**

1. In the General section, add `C:\MAK\makRti5.0\lib` to the Additional Library Directories property.
2. In the Input section, on the Additional Dependencies property, for release builds, add:

```
libRTI-NG.lib libFedTime.lib ws2_32.lib netapi32.lib  
comctl32.lib
```

For debug builds, add:

```
libRTI-NGd.lib libFedTimed.lib ws2_32.lib netapi32.lib  
comctl32.lib
```

3. In the Input category, on the Ignore Specific Library property, for release builds, add `msvcrt.lib`. For debug builds, add `msvcrt.lib`.

3.3. Compiling and Linking with HLA 1516

► **To build an application that uses the MAK RTI 1516:**

1. Include `RTI1516.h` in any source files that use RTI functionality.
2. Add the MAK RTI 1516 `./include/HLA1516` and `./lib` directories to the appropriate search paths.
3. Link with the MAK RTI 1516 libraries.

3.3.1 Compiling and Linking on Linux

If you have installed the MAK RTI in, for example, `/usr/products/makRti5.0`, this should display on your compile line:

```
-I/usr/products/makRti5.0/include/HLA1516
```

and this should display on your link line:

```
-L/usr/products/makRti5.0/lib -lrti1516 -lfedtime1516
```

3.3.2 Compiling and Linking on Windows

Assume that the MAK RTI is installed in `C:\MAK\makRti5.0`. (The Microsoft Visual C++ user interfaces vary slightly from one version to another, so the following description is somewhat genericized.)

► **To compile for HLA 1516, on the C/C++ section of the Project Property Pages:**

1. In the General section, add `C:\MAK\makRti5.0\include\HLA1516` to the Additional Include Directories property.
2. For MAK RTI plug-ins, add `C:\MAK\makRti5.0\include\MAK` to the Additional Include Directories property.
3. In the Code Generation section, in the Run-time Library property, select Multithreaded DLL (/MD) for release builds. Select Multithreaded Debug DLL (/MDd) for debug builds.

► **To link to the HLA 1516 libraries, on the Linker section of the Project Property Pages dialog box:**

1. In the General section, add `C:\MAK\makRti5.0\lib` to the Additional Library Directories property.
2. In the Input section, on the Additional Dependencies property, for release builds, add:

`librti1516.lib libfedtime1516.lib ws2_32.lib netapi32.lib
comctl32.lib`

For debug builds, add:

`librti1516d.lib libfedtime1516d.lib ws2_32.lib netapi32.lib
comctl32.lib`

3. In the Input category, on the Ignore Specific Library property, for release builds, add `msvcrt.lib`. For debug builds, add `msvcrt.lib`.

3.4. Compiling and Linking with HLA Evolved

To compile and link federates with HLA Evolved, follow the instructions for compiling and linking with HLA 1516, except that you use the HLA Evolved directories and libraries instead of the HLA 1516 directories and libraries. (Change 1516 to 1516e and HLA1516 to HLA1516E.)

See "3.3. Compiling and Linking with HLA 1516" on the previous page for the instructions.

3.5. Compiling and Linking with HLA 4

To compile and link federates with HLA 4, follow the instructions for compiling and linking with HLA 1516 except that you use the HLA 4 directories and libraries instead of the HLA 1516 directories and libraries.

The HLA 4 include directory is HLA1516_2025. The HLA 4 libraries are named following the guidance of SISO-REF-075-2023 standard. The MAK RTI only supports 64-bit libraries.

3.5.1 Compiling and Linking on Linux

The MAK RTI supports Red Hat 8 and Red Hat 9 for Linux platforms, which use the GCC compiler version 8 and version 11 respectively.

For RedHat 8 applications linking to MAK RTI HLA 4, the following should appear on your link line:

```
-L/usr/products/makRti4.6/lib -lrti1516_2025gcc8 -lfedtime1516_2025gcc8
```

For RedHat 9 applications linking to MAK RTI HLA 4, the following should appear on your link line:

```
-L/usr/products/makRti4.6/lib -lrti1516_2025gcc11 -lfedtime1516_2025gcc11
```

3.5.2 Compiling and Linking on Windows

The MAK RTI Windows platform support uses the Visual Studio 2017 compiler version.

For Windows applications linking to MAK RTI HLA 4, add the following to the Input section on the Additional Dependencies property in your project.

For release builds, add:

```
librti1516_2025vc141.lib libfedtime1516_2025vc141.lib ws2_32.lib  
netapi32.lib comctl32.lib
```

For debug builds, add:

```
librti1516_2025vc141d.lib libfedtime1516_2025vc141d.lib ws2_32.lib  
netapi32.lib comctl32.lib
```

3.6. Federation Time

According to the HLA specifications, the RTI requires all federations to supply an implementation of time. The RTI API also requires each federate to link with a time library even if it does not use the time management services. The MAK RTI provides default time implementation libraries to help generate federate executables. The implementations can also be used for federation execution. The RTI API allows a federate to link using any time implementation library and then later execute the federate in a federation using a different time implementation library. If you are not using the time management services it does not matter which library is used during execution.

3.6.1 The HLA 1.3 Time Implementation

The RTI 1.3 API requires that all federates supply an implementation of the logical time abstraction in a library called *libfedtime*. The default implementation provided in the MAK RTI contains a derivation of `RTI::FedTime` called *RTIfedTime*, which is defined in *fedtime.hh*.

In addition to the *RTIfedTime* class, the library contains an implementation of the abstraction's destructor `RTI::FedTime::~~FedTime` and the factory `RTI::FedTimeFactory`. The factory class is primarily for use by the RTI so that it can create instances of `RTI::FedTime` and *LogicalTimeInterval*. When a federate uses the time management services, it should use one of the concrete logical time derivations directly. The derivations provide constructors and methods that allow the federate to work with the logical time abstraction using the native C++ type (double).

3.6.2 The RTI 1516 Time Implementation

The RTI 1516 API requires that all federates supply an implementation of the logical time abstraction in a library called *libfedtime1516*. The default logical time library supplied with the MAK RTI implements these concrete derivations of the logical time abstraction:

- An implementation based on integer representation.
- An implementation based on floating point representation.
- A deprecated implementation is the default. It duplicates the floating point representation.

You can choose the time implementation used by the RTI when you create the federation. In the `createFederationExecution` service call, the third parameter is an optional string that specifies the time implementation to use. To specify the floating point representation, pass in `RTI_Float64Time`. To specify the integer representation, pass in `RTI_Integer64Time`. If no time implementation string is specified, the floating point representation is used.

These classes are provided to support the logical time derivations:

- Integer: *LogicalTimeFactoryImplInteger*, *LogicalTimeImplInteger*, and *LogicalTimeIntervalImplInteger*.
- Floating point: *LogicalTimeFactoryImplFloat*, *LogicalTimeImplFloat*, and *LogicalTimeIntervalImplFloat*.
- Deprecated Floating point: *LogicalTimeFactoryImpl*, *LogicalTimeImpl*, and *LogicalTimeIntervalImpl*.

In addition to these classes, the library contains a *LogicalTimeFactoryFactory*. The factory-factory and factory classes are primarily for use by the RTI so that it can create the appropriate logical time factory and use the factory to create instances of *LogicalTime* and *LogicalTimeInterval*. When a federate uses the time management services, it should use one of the concrete logical time derivations directly. The derivations provide constructors and methods that allow the federate to work with the logical time abstraction using the native C++ types (double and unsigned integer).

The class declarations are contained in files that have the same name as the class and end in *.h*. They are in *./include/HLA1516/RTI*.

3.6.3 The HLA Evolved Time Implementation

The RTI HLA Evolved API requires that all federates supply a library called *libfedtime1516e* which contains the implementation for the *LogicalTimeFactoryFactory* class. The factory-factory and factory classes are primarily for use by the RTI so that it can create the appropriate logical time factory and use the factory to create instances of *LogicalTime* and *LogicalTimeInterval*.

The HLA Evolved API also requires that the RTI provide an implementation of the class *HLAlogicalTimeFactoryFactory*. This factory can be used to create factories for the two standardized time implementations, integer and floating point, which are also supplied by the RTI. The default logical time library supplied with the MAK RTI contains a *LogicalTimeFactoryFactory* that simply forwards all time factory creation requests to the *HLAlogicalTimeFactoryFactory*.

You can choose the time implementation used by the RTI when you create the federation. In the `createFederationExecution` service call, the last parameter is an optional string that specifies the time implementation to use. To specify the floating point representation, pass in `HLAfloat64Time`. To specify the integer representation, pass in `HLAinteger64Time`. If no time implementation string is specified, the floating point representation is used.

These classes are defined in the HLA Evolved API to support the logical time derivations:

- Integer: *HLAinteger64TimeFactory*, *HLAinteger64Time*, and *HLAinteger64Interval*
- Floating point: *HLAfloat64TimeFactory*, *HLAfloat64Time*, and *HLAfloat64Interval*

When a federate uses the time management services, it should use one of the concrete logical time derivations directly. The derivations provide constructors and methods that allow the federate to work with the logical time abstraction using the native C++ types (double and unsigned integer).

The class declarations are contained in files that have the same name as the class and end in *.h*. They are in *./include/HLA1516E/RTI/time*.

3.6.4 The HLA 4 Time Implementation

The RTI HLA 4 Time implementation provides the same two standardized time implementations, integer and floating point, as the HLA Evolved API. The HLA 4 header files for time related classes have been reorganized to better reflect the RTI library where the implementation is defined.

The *LogicalTimeFactoryFactory* class is now declared in its own separate header file located in the *./include/HLA1516_2025/RTI/libfedtime* folder. The classes responsible for the RTIs provided time implementation are now located in the *./include/HLA1516_2025/RTI/time* folder. This folder includes header files for the following classes:

- Integer: *HLAinteger64TimeFactory*, *HLAinteger64Interval*, *HLAinteger64Time*
- Floating Point: *HLAfloat64TimeFactory*, *HLAfloat64Interval*, *HLAfloat64Time*
- Base : *LogicalTimeInterval*, *LogicalTime*
- Factory: *HLAlogicalTimeFactoryFactory*, *LogicalTimeFactory*

3.7. Java Bindings for the MAK RTI

The MAK RTI provides a set of Java bindings. The Java binding is a thin layer of C++ code that exposes the native C++ API of the RTI to Java applications.



4. RTI Settings

This chapter provides general information about editing the RTI settings. For descriptions of the settings, please see the comments in the *rid.mtl* file, the sections in this manual that explain how to configure the MAK RTI, and the "Appendix A. RTI Settings Reference" on page 203.

4.1. The RTI Initialization Data (RID) File	42
4.2. Selecting the RTI Settings to Use	42
4.2.1 Search Order for RID Files	42
4.3. Editing RTI Settings	43
4.3.1 Adding Custom Configurations	57
4.3.2 Importing RTI Settings	58
4.3.3 Deleting RTI Settings Configurations	58
4.4. Displaying Configuration Settings in the Federations View	59
4.5. Configuring HLA Switch Values	60
4.6. Configuring Use of Advisory Messages	60
4.7. Consistency of RTI Settings	60
4.8. Specifying RTI Settings Programmatically in HLA 1516	61
4.9. The RTI Settings File Consistency Checker	62
4.9.1 Enabling RTI Settings Consistency Checking	62
4.9.2 Selecting the Parameters to be Checked	63

4.1. The RTI Initialization Data (RID) File

The RTI initialization data (RID) file, also called the RTI settings file, is the configuration file for an RTI. In the MAK RTI RTI settings file, you can set parameters that control the RTI's networking configuration, enable or disable services, and tune performance. The `rtiexec`, RTI Forwarder, and each federate's Local RTI Component read the RTI settings file when they are initialized. For a description of the RTI settings file format and file parameters, see "Appendix A. RTI Settings Reference" on page 203.

4.2. Selecting the RTI Settings to Use

The default RTI settings file is `rid.mtl`. The RTI also includes several RTI Settings configurations, which are variations of `rid.mtl`. You can edit these configurations and you can create your own RTI Settings configurations. (For details about creating and editing RTI Settings configurations, see "4.3. Editing RTI Settings" on the facing page.)

When you run an HLA application, if an RTI forwarder is not running on the local computer, you are prompted to select an RTI Connection. As part of specifying a connection, you can select the RTI settings that you want to use. Your options are "Use Existing Configuration" or any of the RTI Settings configured in the RTI Assistant.

If you choose Use Existing Configuration, the RTI reads the RTI Settings file specified by the `RTI_RID_FILE` environment variable. If you select a specific RTI Settings configuration, it uses the values in that configuration. (For details about the `RTI_RID_File` environment variable, see "4.2.1 Search Order for RID Files" below.)

You create and edit RTI Settings configurations in the RTI Assistant. It lets you edit the most commonly used settings. Other settings can only be changed by editing a settings file using a text editor. In this manual, when we discuss a particular setting, if it can be edited in the RTI Assistant, we use the name of that setting as represented in the Assistant. If the setting can only be edited in a text editor, we use the name of the parameter. Table 2 on page 45 cross references the settings in the Assistant to the parameter names in the RTI settings file. For details about editing RTI settings, see "4.3. Editing RTI Settings" on the facing page.



- The RTI settings file is not required for basic RTI use. Use it if you want to change the default parameter values.
- The connection configurations for the RTI Assistant override the RTI settings file parameters that specify RTI connection values unless you force the RTI to use the RTI settings file parameters. For details, please see "[Managing Connection Configurations](#)" in the *MAK RTI User Guide*.

4.2.1 Search Order for RID Files

The search order for the RTI Settings file depends on how you specify it. If you do not specify the `RTI_RID_FILE` environment variable, the RTI searches for `rid.mtl` as described in this section. If you specify `RTI_RID_FILE`, you can use a configuration file with a name other than `rid.mtl`. (It must use the same MTL syntax as `rid.mtl`.) In HLA 1516 (but not HLA Evolved), you can override `RTI_RID_FILE` by passing a wstring to the `createRTIambassador` call. (For details, see "4.8. Specifying RTI Settings Programmatically in HLA 1516" on page 61.)

 On Windows, you can use the RTI Chooser to specify the RID file to use with a particular version of the RTI. The RTI Chooser takes care of setting the environment so that the correct RID file is used. For details about using the RTI Chooser, please see "[Configuring the MAK RTI Version and RTI Settings to Use](#)" in the *MAK RTI User Guide*.

► **If the `RTI_RID_FILE` environment variable is not defined, the search order is:**

1. Try to open *rid.mtl* in the directory from which you run the application (that is, the working directory, which is not necessarily the directory in which the executable is located).
2. Try to open the file specified by *RTI_CONFIG/rid.mtl*.
3. If a RID file is not found, print a warning and open using default values.

► **If the `RTI_RID_FILE` environment variable is defined, the search order is:**

1. Try to open the file specified in `RTI_RID_FILE`.
2. Try to open the file specified by *RTI_CONFIG/RTI_RID_FILE*.
3. If a RID file is not found, the MAK RTI stops looking for the RID file and fails.

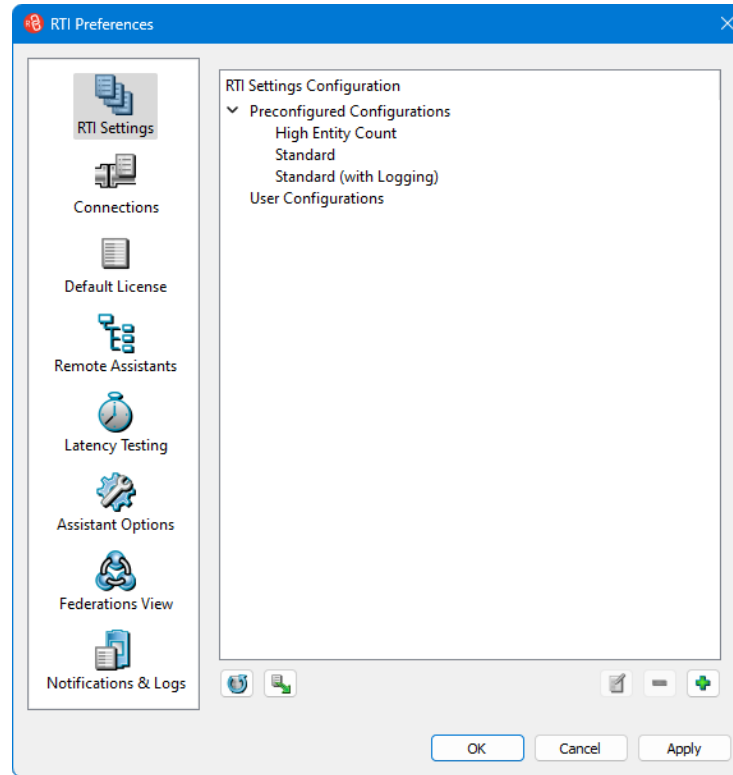
4.3. Editing RTI Settings

You can edit RTI settings in a text editor or using the RTI Assistant. (The RTI Assistant lets you edit the most commonly changed settings. Some settings can only be edited using a text editor.) This section explains how to edit them using the RTI Assistant. For a list of RTI settings and what they mean, see the "Appendix A. RTI Settings Reference" on page 203.

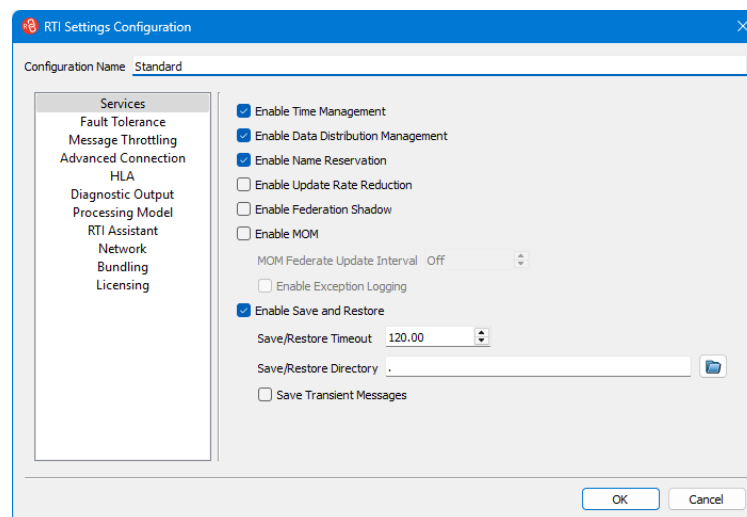
By default, the RTI Assistant includes one RTI settings configuration, called Standard. It is based on *rid.mtl*. The first time that you edit the standard configuration, it saves the settings to a new settings file. Thereafter, it saves all changes to that file. On Windows, RTI Settings files are saved to `%APPDATA%\MAK\RTI\5.0\RIDS`. On Linux, files are saved to `~/.mak/RTI/5.0/RIDS`. The new files include the full set of RTI settings, using default values for settings that are not configurable in the RTI Assistant.

► **To edit RTI Settings:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1). It lists RTI settings configurations provided by MAK and any configurations that you may have created.

Figure 1. RTI Settings page

5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens (Figure 2). It has a set of pages with configuration settings.

Figure 2. RTI Settings Configuration dialog box

7. Edit the settings as desired. Table 2 lists the settings, by page, and cross-references them to the RTI settings file parameters that they represent. For complete descriptions of the parameters, see the "Appendix A. RTI Settings Reference" on

page 203.

8. Click OK.

Table 2. RTI settings

Setting	Parameter
Services	
Enable Time Management	Enables or disables time management services. Default: Disabled. For more information, see "11.1.2 Configuring Time Management Services" on page 162. See also RTI_timeMgmt in Table 12 on page 206.
Enable Data Distribution Management	Enables or disables data distribution management (DDM) services. Default: Enabled. For more information, see "12.1. Configuring DDM Services" on page 168. See also RTI_dataDistMgmt in Table 12.
Enable Name Reservation	Enables or disables the name reservation service in HLA 1516 or HLA Evolved. Default: Enabled. See RTI_enableNameReservation in Table 12.
Enable Update Rate Reduction	Enables or disables update rate reduction. Default: Disabled. See RTI_enableUpdateRateReduction in Table 12.
Enable MOM	Enables or disables management object model (MOM) services. Default: Disabled. For more information, see "11.1.1 Configuring MOM Services" on page 160 See also RTI_momServiceAvailable in Table 12.
MOM Federate Update Interval	Specifies the default interval in seconds at which the MOM Federate object will be updated. Default: Off (no updates sent). See RTI_momFederateUpdateInterval in Table 13 on page 218
Enable Exception Logging	Enables or disables MOM exception logging. Default: Disabled. See RTI_momExceptionLogging in Table 13.
Enable Save and Restore	Enables or disables save/restore when you are using the rtiexec. Default: Enabled See RTI_enableSaveRestoreWhenUsingRtiexec in Table 12 on page 206.
Save/Restore Timeout	Specifies the number of seconds the federation execution waits for a save/restore operation to complete before indicating a failure. Default: 120 seconds. For more information, see "6.9. Configuring Save/Restore" on page 97. See also RTI_saveRestoreTimeout in Table 12.

Table 2. RTI settings (continued)

Setting	Parameter
Save/Restore Directory	Specifies the directory used for writing and reading save files. If the saved federation contains multiple federates of the same type, each federate of a given type must have access to all of the LRC save files of that type. The RTI determines the save files used for each federate. Default: ".". See RTI_saveRestoreDirectory in Table 13 on page 218.
Save Transient Messages	Enables or disables saving of transient messages during a save, to be delivered upon restore. Default: Disabled. For more information, see "6.9. Configuring Save/Restore" on page 97. See "A.3. Federation-Wide Parameters" on page 206 in Table 12 on page 206.
Fault Tolerance	
Enable Federate Heartbeat	Enables or disables the requirement that federates send a heartbeat. Default: Disabled See RTI_enableFederateHeartbeat in Table 12.
Heartbeat Interval	Specifies the frequency, in seconds, with which each federate LRC sends a heartbeat message. Default: 10. Valid only if Enable Federate Heartbeat is enabled. For more information, see "6.8.1 Responding to Abnormal Termination" on page 91. See also RTI_federateHeartbeatInterval in Table 12.
Timeout Interval	Specifies the time period in which the federation must receive a heartbeat or it will forcibly remove the federate. Default: 25. Valid only if Enable Federate Heartbeat is enabled. For more information, see "6.8.1 Responding to Abnormal Termination" on page 91. See also RTI_federateTimeoutInterval in Table 12.
Enable Reconnections	Specifies that the LRC try to reconnect to the RTI Forwarder "Federate Reconnect Pause" seconds after a TCP connection is lost. Default: Disabled. For more information, see "6.8.2 Reconnecting to the RTI Forwarder" on page 94. See also RTI_reconnectEnabled in Table 12.
Federate Reconnect Pause	If Enable Reconnections is enabled, specifies the amount of time, in seconds, to wait before trying to reconnect to the RTI Forwarder. Default: 5. For more information, see "6.8.2 Reconnecting to the RTI Forwarder" on page 94. See also RTI_federateReconnectPause in Table 12.

Table 2. RTI settings (continued)

Setting	Parameter
Rtiexec Reconnect Pause	Amount of time rtiexec waits to reconnect when it loses its TCP connection. This allows the TCP socket to clear. It should be less than the Federate Reconnect Pause.
Automatically Delete Orphans	Specifies whether or not the objects for which a terminated federate owns the privilegeToDelete will be automatically deleted. Default: Enabled. For more information, see "Handling Orphaned Attributes" on page 92. See also "A.3. Federation-Wide Parameters" on page 206 in Table 12.
Message Throttling	
Enable Message Throttling	Enables or disables throttling of transmission and receipt of internal messages (for all versions of the RTI). This parameter also enables or disables buffering of provideAttributeValueUpdate callbacks in HLA 1516 or HLA Evolved. Default: Disabled. See RTI_enableMessageThrottling in Table 13 on page 218.
Transmit Delay	If Enable Message Throttling is enabled, this parameter configures the time to delay before processing requests. Default: Disabled. For more information, see "11.2.1 Configuring the Delay Time for Processing Internal Messages" on page 163. See "RTI_transmitDelay" on page 228 in Table 13.
Transmit Rate	If Enable Message Throttling is enabled, Transmit Rate specifies the number of messages per second that you want to send. Default: Disabled. For more information, see "11.2.2 Transmitting Internal Messages Across Ticks" on page 164. See also RTI_transmitRate in Table 13.
Discovery Message Delay	Delays the sending of discovery messages for the specified number of seconds. This allows discovery messages to be bundled together. Bundling can reduce traffic and processing time if a federate is creating a large number of objects at once. Default: 0.0 seconds. See RTI_discoveryMsgBundlingDelay in Table 12 on page 206.
Auto Provide Delay	If Enable Message Throttling is enabled, enables buffering of automatic provideAttributeValueUpdate callbacks in HLA 1516 or HLA Evolved to limit the number of provides requested if multiple federates require updates. Default: No delay. For more information, see "9.11. Buffering Automatic Update Requests (1516 Specifications Only)" on page 131. See also RTI_autoProvideDelay in Table 13 on page 218.

Table 2. RTI settings (continued)

Setting	Parameter
All Provides Delayed	Specifies that Provide Delay should apply to all update requests, not just those generated due to the autoProvide switch. Message throttling must be enabled and Auto Provide Delay must be set to a non-zero value. Default: Disabled. See RTI_allProvideUpdateRequestsDelayed in Table 13.
Advanced Connection	
Send Internal RTI Messages Reliable (TCP)	Specifies whether or not you want to send the RTI's internal bookkeeping messages using reliable transport. This parameter is ignored if RTI_useRtiExec is set to 0. Default: Enabled. For more information, see "9.1. Choosing Best Effort or Reliable Transport" on page 122. See also RTI_internalMsgReliableWhenUsingRtiexec in Table 12 on page 206.
Use Smart Forwarding with Internal Messages	Enables or disable smart forwarding transport for internal and/or FOM messages (for example, update, reflect, delete, and interactions). Requires the rtiexec process. Ignored if RTI_useRtiExec is not set to 1. All FOM data is sent best effort when FOM reliable is disabled. This should be set to true for most situations as it greatly reduces the number of messages in a federation. Default: Enabled. See RTI_internalMsgSmartForwardingWhenUsingRtiexec in Table 12.
Forwarder Listens to Local UDP Messages	Control whether RTI Forwarder listens to local UDP multicast traffic to distribute to other RTI Forwarders. If selected, this is the equivalent to setting RTI_distributedUdpForwarderMode to 2 (combined mode). See RTI_distributedUdpForwarderMode in Table 12.
Response Interval	Specifies the amount of time (in seconds) that the local RTI component waits to receive a response from the rtiexec when processing the createFederation, destroyFederation, or joinFederation service calls. Default: 3 seconds. For more information, see "6.8.3 Specifying the Response Interval" on page 95. See also RTI_responseInterval in Table 12.

Table 2. RTI settings (continued)

Setting	Parameter
Smart Forwarding Level	Specifies the sender-side filtering mode, where: No subscription-based routing or filtering. Enable subscription based routing. Enable subscription based routing and disable send for updates and interactions that none of the federates have subscribed to. Default: no subscription-based routing or filtering. For more information, see "9.5. Sender-Side Filtering (Smart TCP Forwarding)" on page 124. See also <code>RTI_smartForwardingLevel</code> in Table 12.
FOM Data Transport Type	This parameter sets the FOM Data Transport Type Controller as follows: FOM data is sent either best effort or reliable based on FOM settings. This is the default. It is forced when forceFullCompliance is enabled. All FOM data is sent best effort. It is the forced setting when using lightweight configurations. All FOM data is sent reliable. This is permitted only when <code>RTI_configureConnectionWithRid</code> is 1. See <code>RTI_fomDataTransportTypeControl</code> in Table 12.
Max Forwarder Queue	Specifies the size of the queue for messages that exceed the size of the receive buffer. When a federate does not tick fast enough and its receive buffer fills, it starts placing the messages into the queue. Once the queue has reached the maximum size, the federate is dropped. If the size is set to -1 (default), it queues messages indefinitely until the federate processes them all, or until all system memory is completely consumed. See <code>RTI_maxForwarderQueue</code> in Table 13 on page 218.
Max Number of Federates	Specifies the maximum number of federates allowed in a federation at any one time. This parameter only applies when sender-side filtering is enabled. Each update and interaction message has a bitmask of at least this number of bits appended, so increasing the number of federates increases the overhead. Default: 511. See <code>RTI_maxNumFederates</code> in Table 12 on page 206.
Max Objects Per Federate	The maximum number of objects that a federate can instantiate. Default: 8388607. See <code>RTI_maxObjectsPerFederate</code> in Table 12.

Table 2. RTI settings (continued)

Setting	Parameter
HLA	
Extend Compatibility Between HLA Versions	Extends interoperability between 1.3 and the 1516 versions (for example, handling DDM dimensions). Default: Enabled. For more information, see "1.4.1 RTI Interoperability" on page 23. See also RTI_extend13and1516interop in Table 12.
Relax FOM Consistency Checking	If cleared, configures the RTI to strictly check the format of the FDD file (for example, to ensure presence of transport and order or disallow unknown tags). An error in the FDD file causes the create federation service to fail. This parameter affects the HLA 1516 or HLA Evolved FDD files. If selected, release FOM checking. Default: Enabled (do not use strict FOM consistency checking). See RTI_strictFomChecking in Table 13 on page 218.
Relax Attribute Ownership Unavailable	If cleared, forces a strict attribute ownership unavailable callback. In HLA Evolved, enabling this parameter may cause attributeOwnershipUnavailable to not occur in federates with pending acquisitions in response to the cancelNegotiatedAttributeOwnershipDivestiture and confirmDivestiture service calls when the federate does not receive the requested attributes. Default: Enabled (do not use strict attribute ownership callbacks). See RTI_strictHLA1516eAttributeOwnershipUnavailableCallback in Table 12 on page 206.
Override FED Switches	Specifies which set of switch tables to use. Enabled: Use the RID parameters. Disabled: Use the switch tables defined in the FED file. Force full compliance forces use of the FED file switch table. Default: Enabled. See RTI_preferRidToFedSwitchTable in Table 13 on page 218.
Auto-Provide	Specifies whether the Auto Provide switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_autoProvideSwitch in Table 13.
Convey Region Designator	Specifies whether the Convey Region Designator switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_conveyRegionDesignatorSwitch in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
Convey Producing Federate	Specifies whether the Convey Producing Federate switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Enabled. See RTI_conveyProducingFederateSwitch in Table 13.
Service Reporting	Specifies whether the Service Reporting switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_serviceReportingSwitch in Table 13.
Exception Reporting	Specifies whether the Exception Reporting switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_exceptionReportingSwitch in Table 13.
Delay Subscription Evaluation	Specifies whether the Delay Subscription Evaluation switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_delaySubscriptionEvaluationSwitch in Table 13.
Attribute Scope Advisory	Specifies whether the Attribute Scope Advisory switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See RTI_attributeScopeAdvisorySwitch in Table 13.
Attribute Relevance Advisory	Specifies whether the Attribute Relevance Advisory switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Disabled. See "RTI_attributeRelevanceAdvisorySwitch" on page 221 in Table 13.
Object Class Relevance Advisory	Specifies whether the Object Class Relevance Advisory switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Enabled. See RTI_objectClassRelevanceAdvisorySwitch in Table 13.
Interaction Relevance Advisory	Specifies whether the Interaction Relevance Advisory switch is enabled or disabled. Override FED Switches must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: Enabled. See RTI_interactionRelevanceAdvisorySwitch in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
Diagnostic Output	
LRC Minidumper Level	Configures whether the RTI installs hooks to create a dump file if it crashes. Options are: • No Dump. Do not install hooks. • Minidump. Prompt to create a dump file. • Minidump (No Prompt). • Silent Exit. Exit without prompting or creating a dump file. • Default: No Dump.
Notify Level	Specifies the category of notification messages to print. Default: Info. See RTI_notifyLevel in Table 13 on page 218.
Detach Notify Level	Detaches all notification messages at the given level and above from the standard output. This setting has no affect on logging notifications to a file. Default: Detach All. See RTI_detachNotifyLevelFromStdOut in Table 13.
Log File Directory	Specifies a directory where the log file should be output. Default: "" (the working directory). See RTI_logFileDirectory in Table 13.
LRC Log File Name	Enables logging to a file with the specified name. If just a filename is specified, a file is written for each federate on the computer in the directory from which a federate is executed. If a pathname is specified, data for all federates on the machine is written to the same file unless Reuse Log File is disabled. Default: "makRti.log". For more information, see "7.3.1 Enabling LRC Diagnostic Data Logging" on page 102. See also RTI_logfileName in Table 13.
RTI Exec Log File Name	Enables or disables the rtiexec log file and specifies the file's name. Default: "rtiExec.log". For more information, see "Logging rtiexec Output" on page 67. See also RTI_rtiExecLogFileName in Table 13.
Forwarder Log File Name	Enables or disables the RTI Forwarder log file and specifies the file's name. Default: <i>rtiForwarder.log</i>. See RTI_rtiForwarderLogFileName in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
Reuse Log File	Enables or disables writing logs to the file specified by Log File Name (default: <i>makRti.log</i>). When enabled, it is possible that multiple federates might write to the same log file. If this option is disabled, the logs are written to <i>makRti###.log</i> where <i>###</i> is replaced by the system time in seconds. Each federate will have a different log file. Default: Enabled. For more information, see "Logging rtiexec Output" on page 67. See also <code>RTI_reuseLogFile</code> in Table 13.
Warnings for Disabled Service	Specifies whether or not the RTI Assistant displays a message when an exception is thrown for a disabled service. Default: Enabled. See <code>RTI_enableWarningsForDisabledServices</code> in Table 13.
Exception for Disabled Service	Specifies whether or not function calls throw an exception when they call a service that is not activated in the MAK RTI. Default: Enabled. For more information, see "6.8.5 Choosing Silent Failure Instead of Exceptions" on page 96. See also <code>RTI_throwExceptionForDisabledService</code> in Table 13.
Dump RID	Enables or disables dumping of the contents of the RID file when the notification level is set to verbose (>3). Default: Disabled. See <code>RTI_dumpRid</code> in Table 13.
Dump FOM	If selected, print the FOM to the log file at startup.

Table 2. RTI settings (continued)

Setting	Parameter
Processing Model	
Processing Model	Specifies the synchronous/asynchronous I/O operating mode, as follows: Synchronous mode. Asynchronous I/O. Perform network reads and writes in an asynchronous thread (reduces dependency on tick rate to avoid msg drops). Asynchronous Message Processing. In addition to performing network reads, the asynchronous thread processes messages. However, it will not perform any asynchronous federate ambassador callbacks. The callbacks will be cached and delivered when the federate calls tick or evokeCallbacks. Asynchronous Callbacks. Process messages and invoke callbacks in an asynchronous thread.  For Asynchronous Callbacks (not Asynchronous I/O), the Federate must be able to receive Federate Ambassador callbacks asynchronously (in a thread safe way). Default: Asynchronous I/O. See RTI_processingModel in Table 13 on page 218.
I/O Period	Specifies the amount of time polling operations take, in seconds. Not used in Windows, which is event driven. Must be greater than or equal to zero. Default: 0.1. See RTI_IOPeriod in Table 13.
Tick Wait Period	Determines the smallest increment of time that a tick will cause a federate to wait for messages from the I/O thread when using asynchronous I/O. If set to 0.0, the minimum time specified in the tick service invocation is used instead. Default 0.0. See RTI_tickWaitPeriod in Table 13.
Max I/O Queue	Specifies how many messages can be queued to send before blocking. Default 500000. See RTI_maxIOQueue in Table 13.
Max I/O Count per Tick	Specifies the number of times the thread can cycle before it must yield. Must be greater than zero. Default: 50000. See RTI_maxIOCount in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
RTI Assistant	
Network Statistics Monitoring	When enabled a federate's network statistics can be viewed in the RTI Assistant's RTIspy window. This capability is provided by a plug-in. There are limits on how many plug-ins can be active at the same time. When Network Statistics Monitoring is disabled, all plug-ins should work as expected. When it is enabled, these constraints are in place (meaning plug-ins will not work or will be completely ignored). • Plug-ins cannot override the DtRtiAmbassadorImplementor. • Plug-ins cannot override the DtConnectionMgr. • Plug-ins cannot override the DtFedAmbWrapper. • Default: Enabled. See RTI_enableLRCNetworkStatisticsMonitoring in Table 13 on page 218.
Network Testing Enabled	Enables or disables network testing. Default: Enabled. See RTI_enableNetworkTesting in Table 13.
Network	
TCP No Delay	Allows you to configure the RTI to emphasize minimal latency or maximum throughput. Disabled: Trade latency for throughput. Enabled: Set the TCP_NODELAY socket option for TCP sockets, yielding minimum latency. Default: Enabled. See RTI_tcpNoDelay in Table 13 on page 218.
TCP Buffer Size	Specifies the size of the buffer into which TCP packets get read. Default: 500000 bytes. For more information, see "9.1. Choosing Best Effort or Reliable Transport" on page 122. See also "A.4. LRC-Specific Parameters" on page 217 in Table 13.
Multicast TTL	Specifies an integer value that gets used as the TimeToLive parameter in the header of multicast packets. Default: 2. See RTI_mcastTtl in Table 13.
Socket Receive Buffer Size	Specifies the size of the TCP and UDP network receive buffers. Increasing the size can increase the RTI's tolerance of peak network loading. Default: 2000000. For more information, see "9.8. Configuring the Network Buffer Sizes" on page 129. See also RTI_socketReceiveBufferSize in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
Socket Send Buffer Size	Specifies the socket send buffer size. Default: 200000. See RTI_socketSendBufferSize in Table 13.
Bundling	
Enable Packet Bundling	Bundles messages together into a single TCP packet or UDP datagram, up to the size limit specified by Packet Bundling Size. Default: Enabled. For more information, see "9.3. Packet Bundling for Reliable and Best Effort Messages" on page 123. See RTI_enablePacketBundling in Table 13 on page 218.
Packet Bundling Size	When packet bundling is enabled, specifies the size of the message bundles for TCP packets and UDP datagrams. The value should be <= MTU of the network. Default: 1400. For more information, see "9.3. Packet Bundling for Reliable and Best Effort Messages" on page 123. See also RTI_packetBundlingSize in Table 13.
Do not wait to send bundled packets	Enable "Do not wait to send bundled packets" when you want the LRC to wait at most the IO period (I/O Period setting) before sending data if no other messages are being queued to fill a bundle. When disabled this prevents the latency overhead of waiting the maximum amount of time for more messages to completely fill a bundle. However, it also may cause messages not to be bundled if the throughput is low. Default: Disabled. See RTI_waitIOPeriodForBundling in Table 13.
Enable TCP Compression	Enables compression of RTI message payloads transmitted over TCP connections. Consistency checking is enforced on this parameter. Default: Disabled. For more information, see "9.6. Compressing RTI Messages" on page 127. See also RTI_enableTcpCompression in Table 13.
TCP Compression Level	Specifies the amount of compression applied to RTI message payloads over TCP connections: • 1 – minimal compression. • 2 through 8 – increasing levels of compression. • 9 – maximum Default: 1. For more information, see "9.6. Compressing RTI Messages" on page 127 See RTI_tcpCompressionLevel in Table 13.

Table 2. RTI settings (continued)

Setting	Parameter
Enable UDP Compression	Enables compression of RTI message payloads transmitted in UDP datagrams. Consistency checking is enforced on this parameter. Default: Disabled. For more information, see "9.6. Compressing RTI Messages" on page 127. See also <code>RTI_enableUdpCompression</code> in Table 13.
UDP Compression Level	Specifies the amount of compression applied to RTI message payloads in UDP datagrams, using the same options as TCP Compression Level. Default: 1. For more information, see "9.6. Compressing RTI Messages" on page 127. See also <code>RTI_udpCompressionLevel</code> in Table 13.
Licensing	
RTI Exec Performs Licensing	Enables or disables license checkout from the <code>rtiexec</code> . The <code>rtiexec</code> will check out a license for each joining federate. Default: Disabled. See <code>RTI_rtiExecPerformsLicensing</code> in Table 13 on page 218.
Force Unlicensed for Two	Forces the MAK RTI to use the unlicensed version without checking for a license. This setting saves time at startup. If you have a licensed version of the MAK RTI, do not skip license checking. Default: Disabled. See <code>RTI_forceUnlicensedForTwo</code> in Table 13.
Disable Unlicensed for Two	Enables or disables unlicensed mode in Windows (even if Force Unlicensed for Two is enabled). When this parameter is enabled, federates must check out a license to run. Default: Disabled. See <code>RTI_disableUnlicensedForTwo</code> in Table 13.

4.3.1 Adding Custom Configurations

The standard RTI setting configuration provided in the RTI Assistant is a subset of the settings in `./rid.mtl`. It lets you edit the settings that MAK thinks you are most likely to want to edit. RTI users often want to have several different settings files for use with different types of exercises. For example, MAK includes `./ddm-rid.mtl`, which has settings customized for use with DDM. You can add your own custom configurations.

► **To add a custom RTI settings configuration:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44).

5. Click the Add button (). The RTI Settings Configuration dialog box opens.
6. In the Configuration Name box, enter a name for this configuration.
7. Edit settings as desired.
8. Click OK. The new configuration is saved and added to the list in the RTI Settings window.

4.3.2 Importing RTI Settings

If you have an RTI settings file that you want to edit using the RTI Assistant, you can import it into the RTI Settings page.

► **To import an RTI settings file:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44).
5. Click the Import Configuration button (). A file chooser dialog box opens.
6. Select the settings file you want to import.
7. Click Open. The RTI Settings Configuration dialog box opens.
8. In the Configuration Name box, enter a name for this configuration.
9. Edit settings as desired.
10. Click OK. The new configuration is saved and added to the list in the RTI Settings window.

4.3.3 Deleting RTI Settings Configurations

When you delete a configuration in the RTI Settings page, you delete the settings file that it references.

► **To delete an RTI Settings configuration:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44).
5. Select the configuration you want to delete.
6. Click the Delete button (). The configuration is deleted from the window and the settings file is deleted.
7. Click OK.

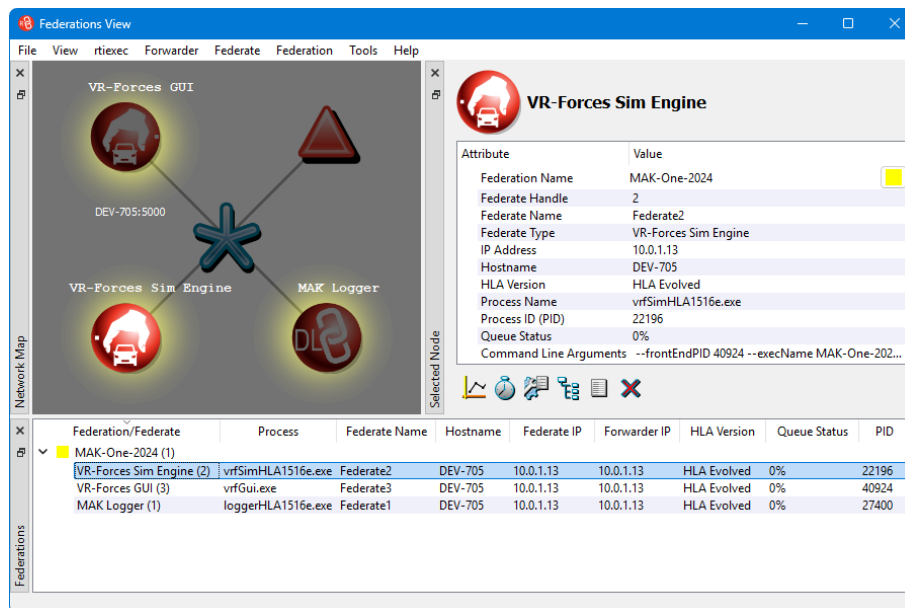
4.4. Displaying Configuration Settings in the Federations View

You can display a list of RID parameters and their settings if you are using an rtiexec connection.

► **To display the RTI's configuration settings:**

1. In the system tray, right-click the RTI Assistant icon (🔧). A context menu opens.
2. Choose **Federations View**. The Federations View window opens (Figure 3).

Figure 3. Federations View

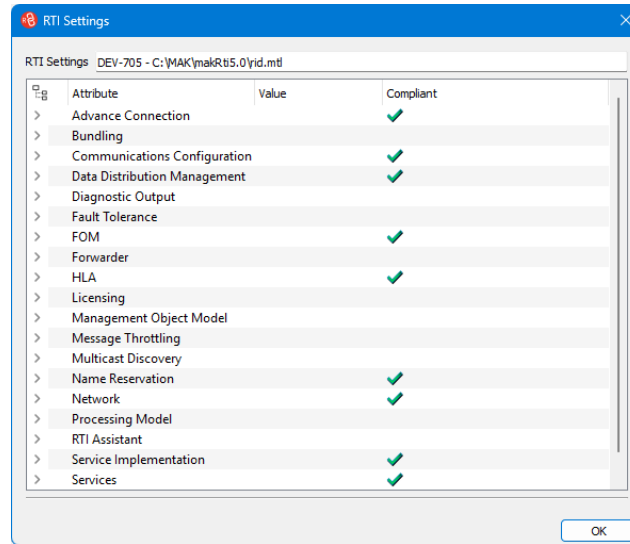


3. Select the Federate whose RID parameters you want to view.
4. Choose **Federate > Federate RTI Settings**, or click the Federate RTI Settings button (🔧). The RID Settings dialog box opens (Figure 4).

The Compliant column indicates compliance of a parameter value:

- A checkmark in the Compliant column means that the current value for that parameter is the correct value for the RTI to be in fully compliant mode.
- An X in the Compliant column means that the parameter is set to a non-compliant value.
- Parameters that do not have a checkmark or exclamation point do not affect compliance.

Figure 4. RID Settings dialog box



4.5. Configuring HLA Switch Values

HLA 1516 and HLA Evolved both define switches in the FDD file that can be used to control various settings of the federation. You can override the FDD file settings with values defined in the RTI settings file. To do so, you must enable the Override FED Switches RTI setting. If force full compliance is enabled, either through the RTI Assistant or via the RTI settings file, Override FED Switches is disabled and the values in the FDD file are used.

4.6. Configuring Use of Advisory Messages

These RTI settings specify the default state for advisory callbacks for classes, attributes, attribute scope, and interactions. Any advisory switches in a 1516 FDD file override these default values. To use any of these parameters, Override FED Switches must be enabled.

- Object Class Relevance Advisory (default: enabled).
- Attribute Relevance Advisory (default: disabled).
- Interaction Relevance Advisory (default: enabled).
- Attribute Scope Advisory (default: disabled).

4.7. Consistency of RTI Settings

The RTI settings files read by the different federates' LRCs and by the rtiexec do not need to be identical. There are many cases where you might want to configure some LRCs differently from others. For example, you might want to have only some LRCs deliver a single callback per tick, or you might want to have different socket buffer sizes from one PC to another. However, some parameters must have the same value for all federates. All RID files must have the same value for **RTI_useRtiExec**. Similarly, all federates in a federation must have the same value for **RTI_udpPort** and **RTI_tcpPort** if they expect to be able to communicate. The tables in "Appendix A. RTI Settings Reference" on page 203

indicate whether a parameter must be identical across the federation or if it can vary from one LRC to another. The RTI can check the consistency of RTI settings files. For details, see "4.9. The RTI Settings File Consistency Checker" on the next page.

4.8. Specifying RTI Settings Programmatically in HLA 1516

The HLA 1516 specification APIs allow configuration parameters to be passed to the RTI Ambassador, though the exact function calls and datatypes vary between HLA 1516, HLA 1516 Evolved, and HLA 4.

- In HLA 1516, the parameters are passed in the RTI Ambassador Factory `createRTIAmbassador` function as a vector.
- In HLA Evolved, the RTI Ambassador's `connect` function takes a single `wstring` parameter.
- In HLA 4, the RTI ambassador's `connect` function takes an `RtiConfiguration` reference parameter.

In all cases, the MAK RTI uses the content of these arguments as a means to modify the RTI settings file parameters processed during initialization of the MAK RTI. The strings can specify the RTI settings file that is initially loaded and additional settings:

- If an RTI settings file name is included, it overrides any other method of specifying the file, such as the `RTI_RID_FILE` environment variable.
- If RTI settings are included, they are applied after reading the settings file (in effect overriding the values of the file). Each string must be a complete lisp expression that conforms to the syntax in the settings file. Only valid lisp expressions are applied to the RTI environment.

This feature allows individual federates to programmatically override the selection of the RTI settings file, individual parameters, or both.

The exact syntax is:

```
RTI_RID_FILE pathToRIDFile # lisp_expression1*
```

where:

- *RTI_RID_FILE* is optional. If included, the `RTI_RID_FILE` must be the first token in the string. The RID text following the token up to the '#' delimiter character is taken as the RID file name. If the # is omitted, then the remainder of the string text is used as the file name.
- *lisp_expression* may contain zero or more lisp expressions.

For HLA 1516, each string in the vector is processed in order.

If *RTI_RID_FILE* is included in the string, the MAK RTI tries to open the specified file. If the file is not found, the configuration process fails without attempting a search for another file. If *RTI_RID_FILE* is omitted, the MAK RTI uses its default search order for determining a global baseline RID file using the `RTI_RID_FILE` or `RTI_CONFIG` environment variables.

For HLA 4, the `RtiConfiguration` class has three properties: Configuration Name, RTI Address, and Additional Settings. The MAK RTI currently supports only the Additional Settings property, which is a single `wstring` parameter similar to the HLA Evolved local settings designator.

4.9. The RTI Settings File Consistency Checker

The MAK RTI can provide runtime verification of RTI settings consistency across all federation executions. The RTI settings file used by the `rtiexec` acts as a master settings file and all other settings files are compared to it. The RTI settings file consistency checker allows you to guarantee that certain parameters are identical throughout the federation. (As noted in "4.7. Consistency of RTI Settings" on page 60, some parameters can vary among federates within a federation.)

Consistency checking has a small cost:

- It increases startup time for all federates, because they must pass messages to the RTI to verify their configuration. If this is a concern, federation managers may want to enable consistency checking during integration, and then disable it once all differences have been resolved.
- Use of consistency checking requires the `rtiexec` and is therefore unavailable in lightweight mode.

 It is strongly recommended that you enable reliable messaging for internal RTI messages (Send Internal RTI Messages Reliable (TCP)) when using consistency checking to guarantee that no consistency check messages are lost.

4.9.1 Enabling RTI Settings Consistency Checking

You can enable and disable consistency checking on a per-federate basis. It is disabled by default. If you enable consistency checking, it must be enabled in the RTI settings file for the `rtiexec` and all federates that want to verify their settings files.

You enable and disable RTI settings consistency checking with the **RTI_**`ridConsistencyChecking` parameter. **RTI_**`ridConsistencyChecking` has these options:

- 0 – Disable consistency checking.
- 1 – Enable consistency checking and automatically correct differences in federate-wide parameter settings.
- 2 – Enable consistency checking and throw a `FederateInternalError` exception if the RID consistency check failed. Setting **RTI_**`ridConsistencyChecking` to 2 is useful for federate developers who want to shut down their federates and correct any inconsistencies before joining the federation.

i The FOM consistency feature relies on the ability to establish a connection between a federate and the tester (presumably the rtiexec), however if the federate's Enable TCP Compression setting and the tester's setting are different, the two cannot communicate, and consequently the FOM consistency test request and response messages cannot be exchanged. The workaround is to ensure that the Enable TCP Compression setting is consistent among all participants prior to launch.

4.9.2 Selecting the Parameters to be Checked

By default the consistency checker tests a large subset of the parameters that must be consistent across the federation in most federations. The default RTI settings file lists the parameters in the default consistency set.

However, you may want to enforce a stricter checking policy. You can change the consistency set using the **RTI-addRidParametersToOverride** and **RTI-removeRidParametersToOverride** parameters in the rtiexec's RTI settings file. Uncomment the parameter you want to use and list each parameter to be added to or removed from the set.

For example, although Enable TCP Compression (**RTI_enableTcpCompression**) and Enable UDP Compression (**RTI_enableUdpCompression**) must be consistent at start-up and cannot be checked for consistency, the TCP Compression Level (**RTI_tcpCompressionLevel**) and UDP Compression Level (**RTI_udpCompressionLevel**) settings can vary within a federation, so they are not checked for consistency by default. Federations that have enabled compression and want to enforce a specific level of compression can add the following to the RID file:

```
(RTI-addRidParametersToOverride
  (list "RTI_tcpCompressionLevel" "RTI_udpCompressionLevel") )
```

 The following RTI settings are evaluated before the FOM consistency check is performed. Therefore, they cannot have their consistency enforced by the consistency checker. It is your responsibility to ensure that, if required, they are the same in all federates.

- **RTI_enableBigMessage**
- **RTI_tcpPort**
- **RTI_udpPort**
- **RTI_enablePacketBundling**
- **RTI_enableTcpCompression**
- **RTI_enableUdpCompression**
- **RTI_internalMsgReliableWhenUsingRtiexec**
- **RTI_mcastDiscoveryEnabled**
- **RTI_ridConsistencyChecking**
- **RTI_tcpForwarderAddr**
- **RTI_useRtiExec.**

Most of these parameters are federation-wide settings and must be the same in all RTI settings files. However, a few such as **RTI_tcpForwarderAddr** may vary if you use distributed forwarders.



5. RTI Applications

This chapter describes the `rtiexec` and applications included with the MAK RTI that help customers better understand or use the HLA environment.

5.1. The <code>rtiexec</code>	66
5.1.1 Configuring the <code>rtiexec</code>	66
5.1.2 Configuring Diagnostic Messages	67
5.2. The RTI Assistant	67
5.2.1 Specifying the RTI Assistant's Log Update Interval	68
5.2.2 Shutting Down the RTI Assistant	69
5.2.3 Configuring the Display of RTI Assistant Messages	69
5.2.4 Connecting to Remote Assistants	69
5.2.5 Specifying the Warning Tick Interval	71
5.2.6 Common Operations in RTI Assistant Log Windows	71
5.2.7 Specifying the RTI Assistant's Ports	72
5.2.8 Specifying the RTI Assistant's Multicast Address	73
5.2.9 Specifying the RTI Assistant's Multicast Time-to-Live	73
5.2.10 Disabling the RTI Assistant	73
5.2.11 RTI Assistant Command-Line Options	73
5.3. The RTI Forwarder Application	74
5.3.1 Starting an RTI Forwarder	74
5.3.2 Shutting Down an RTI Forwarder	77
5.4. The RTI Federate Protocol Server	77
5.4.1 Starting the RTI Federate Protocol Server	78
5.5. The Federation Shadow Application	78
5.5.1 RTI Federation Shadow Command-Line Options	79
5.6. The RTI Command Line Tool	80
5.6.1 Listing Federation Executions and Federates	82
5.6.2 Deleting Federates	82

5.1. The rtiexec

The rtiexec implements any centralized functionality required by the RTI. For example, the rtiexec keeps track of which federates are joined to each federation execution, assigns federate handles, keeps track of which federates have not yet achieved synchronization points, computes LBTS (Lower Bound Time Stamps) for time management, orchestrates saves and restores, and so on. To use the MAK RTI in its fully compliant mode, you must run the rtiexec before you start any federates.

You must use the rtiexec if your federation relies on any of the following:

- Reliable transport (TCP)
- Time management
- Synchronization points
- Federation save and restore
- The RTI Assistant network map

Other RTI services (including MOM, DDM, and ownership management) work in lightweight mode, but in certain cases functionality is limited. For more information, see "Section 6.3.1 Limitations of Lightweight Mode" on page 86.

If you are not using any of the services that require the rtiexec, then the choice of whether or not to use it is a trade-off between the convenience of lightweight mode, and the full compliance and robustness of running with the rtiexec. In general, you are not giving up performance by using the rtiexec. Object attribute updates and interactions sent with best effort transport always get exchanged directly among the federates without going through the rtiexec. Furthermore, while the rtiexec is often associated with reliable message forwarding, reliable forwarding is a separate performance issue. The RTI can be configured to use best effort transport for internal messages or a separate forwarder process can perform the forwarding of reliable messages. (For more information, see "5.3. The RTI Forwarder Application" on page 74.) Either way, it is not the rtiexec processing that is an issue.

The rtiexec is required for just a few services that are best processed using a centralized approach. For most federations, the only time that the rtiexec's centralized processing affects a federate is during create, destroy, join, and resign. During most of your federation, it neither sends nor processes any messages.

 If you use the rtiexec, internal message reliable mode is recommended, rather than best effort.

5.1.1 Configuring the rtiexec

In addition to the command-line options described in the table in "[Starting the rtiexec](#)" in the *MAK RTI User Guide*, you can configure the rtiexec in an RTI settings configuration. The MAK RTI provides several options for configuring the rtiexec. Among the most important options to consider are the use of best effort or reliable transport, and whether or not you will use packet forwarding. For details, see "5.3. The RTI Forwarder Application" on page 74 and "9.1. Choosing Best Effort or Reliable Transport" on page 122.

5.1.2 Configuring Diagnostic Messages

The MAK RTI can provide a variety of diagnostic data to help you debug problems. In addition to the diagnostic resources referenced in this section, you can use the RTIspy to provide additional diagnostic information. For details, see "Chapter 8. Monitoring the Network" on page 105.

Logging rtiexec Output

You can log rtiexec output to file. The RTI Exec Log File Name setting specifies the name of the log file for the rtiexec process. A unique identifier is inserted in the file name when Reuse Log File is enabled. This parameter is overridden by the `rtiexec -l` option.

Limiting Diagnostic Messages

- To prevent the output of diagnostics, use the `--setNotifyQuiet` (or `-q`) command-line option.

5.2. The RTI Assistant

The RTI Assistant is the user interface for the MAK RTI. It starts automatically when you log in or start a federate (this behavior is configurable) and monitors the status of all RTI-related activity on your computer. It also communicates with RTI Assistants on other computers on the network. The RTI Assistant lets you:

- View federation information.
- View information about your RTI network connections.
- Configure connections.
- Configure some RTI behaviors.
- Configure and run latency testing.
- Resign federates.
- Manage remote assistants.
- Start and shut down the rtiexec.
- Start and shut down RTI forwarders.
- View federate logs and notification history for federates running on the local machine.
- Edit RTI settings.

 The federate-specific features of the RTI Assistant, such as viewing log files and resigning federates, are only available for federates that are running on the local machine.

You can configure these aspects of the RTI Assistant:

- How it starts up. For details, please see "[Configuring the RTI Assistant's Startup Behavior](#)" in the *MAK RTI User Guide*.
- Notification and log behaviors.

- Whether or not to keep running the RTI Assistant after all other RTI components have shut down. For details, please see "[Specifying the RTI Assistant's Exit Policy](#)" in the *MAK RTI User Guide*.
- How to display RTI Assistant messages.
- Lightweight connection persistence.
- The update request interval.

You can clear the default license and connection parameters, and you can set an environment variable to disable the RTI Assistant.

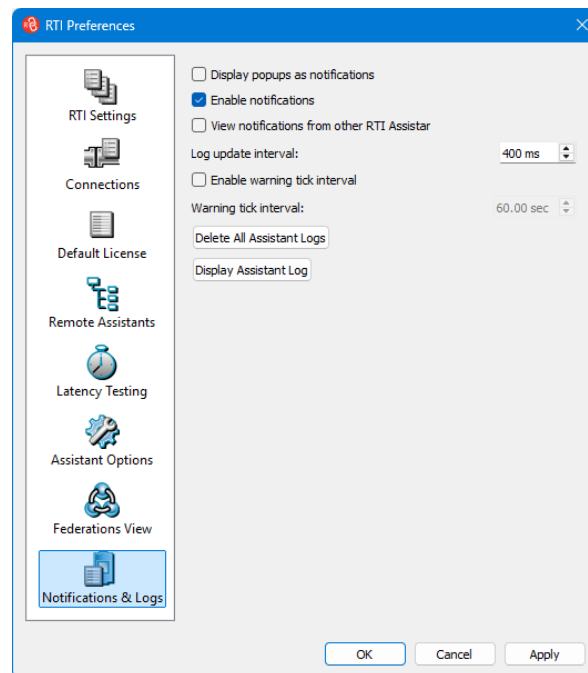
For more general information about the RTI Assistant, please see the [The RTI Assistant](#) chapter in the *MAK RTI User Guide*.

5.2.1 Specifying the RTI Assistant's Log Update Interval

► To specify how frequently the RTI Assistant updates the notification logs:

1. In the system tray, right-click the RTI Assistant icon (🔊). A context menu displays.
2. Choose **Preferences**. The RTI Preferences dialog box opens.
3. Select the Notifications and Logs page (Figure 5).

Figure 5. RTI Preferences dialog box, Notifications and Logs



4. Enter a value, in milliseconds, in the Log Update Interval text box.
5. Click OK.

5.2.2 Shutting Down the RTI Assistant

If you do not configure the RTI Assistant to shut down automatically, you can shut it down manually. When you shut down the RTI Assistant, it forces all LRCs and rtiexecs on the local computer to shut down. It does not affect RTI components on remote computers.

 Shutting down the RTI Assistant in this manner can have undesirable consequences. If remote federates are connected to an rtiexec on the local machine, they will lose their connection.

► **To shut down the RTI Assistant:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose **Shut Down All**.

Disabling the Shut Down All Option

If you do not want RTI users to be able to shut down the RTI Assistant, you can disable the Shut Down All option on the RTI Assistant menu. To do so, add the RTI_ASSISTANT_REMOVE_SHUT_DOWN_ALL environment variable to the user's environment. You do not have to specify a value. The variable just has to be set.

5.2.3 Configuring the Display of RTI Assistant Messages

By default, when an LRC detects a serious problem that is not characterized as a Fatal or Warning message (which would trigger a notification message), it displays a message in a pop-up window in the middle of the screen. You can configure the RTI Assistant to display these messages as notifications (the messages displayed above the RTI Assistant icon).

► **To configure the display of RTI Assistant messages:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose **Preferences**. The RTI Preferences dialog box opens.
3. Select the Notifications and Logs page (Figure 5 on the previous page).
4. Select or clear the Display Popups as Notifications check box.
5. Click OK.

5.2.4 Connecting to Remote Assistants

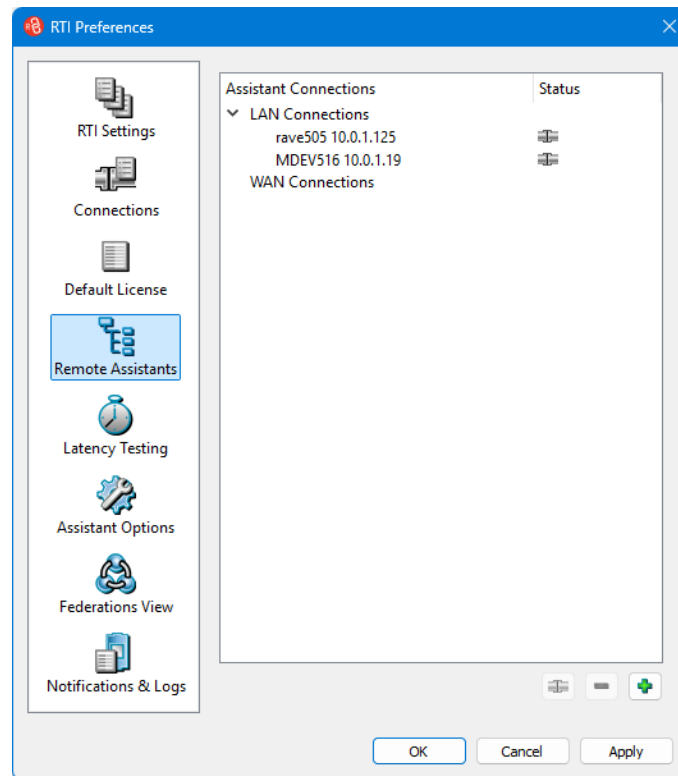
When the RTI Assistant starts, it automatically connects to any other RTI Assistants that it detects on the local network. When you are connected to remote assistants, you can start and stop RTI Forwarders and you can add new RTI connections to the systems they represent.

In addition to automatically connecting to RTI Assistants on your LAN, you can manually connect to RTI Assistant located on VPNs and WANs. Once you are connected, you have the same capability of managing those connections as you do on your LAN. Since connecting to a remote connection essentially means connecting to a remote forwarder, this method of connecting to forwarders is an alternative to using routes files to configure connections. (For information about using routes files to connect to forwarders, see "10.6. Configuring RTI Forwarders for Distributed Forwarding" on page 146.)

► **To add an RTI Assistant connection:**

1. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
2. Choose **Preferences**. The RTI Preferences dialog box opens.
3. Select the Remote Assistants page (Figure 6).

Figure 6. Remote Assistants page



4. Select the LAN Connections line in the window or the WAN Connections line, as desired.
5. Click the Add button. The Assistant Connection dialog box opens.
6. In the Assistant Connection dialog box, enter the IP Address of the computer you want to connect to.
7. Select a protocol in the Protocol list.
8. Specify a port.
9. Click OK. The connection is added to the window.

Managing Remote Assistant Connections

You cannot disconnect from remote assistants that are automatically detected on your LAN. You can connect to and disconnect from remote assistants that you have added manually.

► **To connect to a disconnected remote assistant:**

1. Select the connection in the Remote Assistants list.
2. Click the Connect button () or right-click and choose **Connect**.

Removing a Remote Assistant Connection

► **To remove a remote assistant connection:**

1. Select the connection in the Remote Assistants list.
2. Click the Remove Assistant Connection button () or right-click and choose **Remove Assistant Connection**.

5.2.5 Specifying the Warning Tick Interval

The Warning Tick Interval is the interval at which you would like the RTI Assistant to give you a warning if a federate has not ticked. This allows you to identify federates that are doing so much processing that they cannot keep up the necessary tick rate. If the federate goes too long without ticking, the RTI Assistant displays a message indicating this. You can configure the interval to meet the needs of your federates.

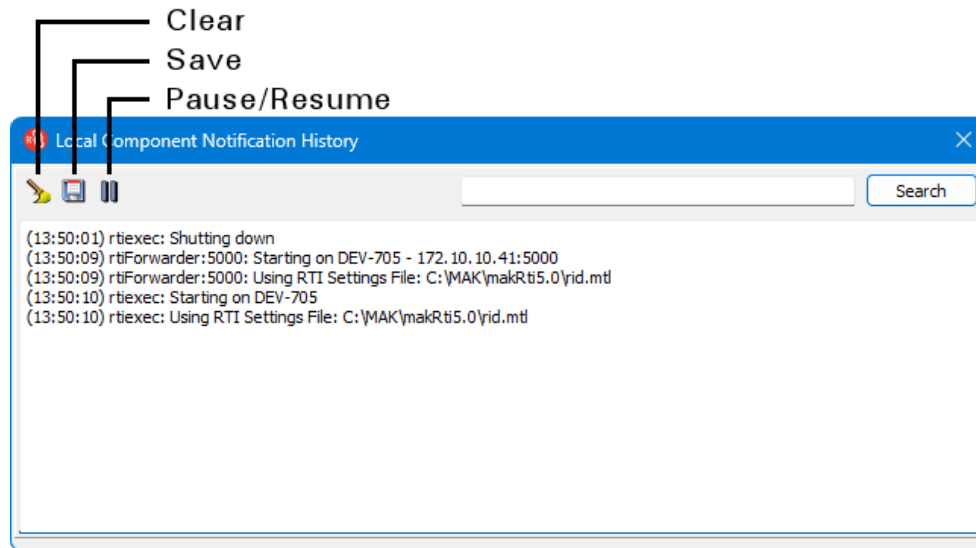
► **To configure the display of RTI Assistant messages:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose **Preferences**. The RTI Preferences dialog box opens.
3. Select the Notifications and Logs page (Figure 5 on page 68).
4. Select the Enable Warning Tick Interval check box.
5. Specify the interval time, in seconds.
6. Click OK.

5.2.6 Common Operations in RTI Assistant Log Windows

The RTI Assistant has several windows for displaying messages: the rtiexec Console, the federate Log window, the Assistant Log, and the Local Component Notification History window. These windows all allow you to pause and resume display of messages, clear the window, and save the log history using icons that are common to the windows (Figure 7). You can also search for words or phrases in the log files.

Figure 7. Console window



 The Assistant Log is primarily for the use of MAK support engineers when they provide customer support.

Pausing and Resuming the Display of Console Messages

If a log or console window is receiving messages faster than you can read them, you can pause the display. While the display is paused, new messages are saved and they are printed to the console when you resume the display of messages.

- To pause or resume display of console or log messages, click the Pause/Resume button (.

Saving Console Output to a File

- To save console or log window output to a file:

1. Click the Save button (.
2. Enter a filename in the Save File dialog box. The RTI saves the current contents of the console.

Clearing the Console Window

- To clear a log or console window, click the Clear button (.

5.2.7 Specifying the RTI Assistant's Ports

The RTI Preferences, Assistant Options page displays the RTI Assistant's peer and client ports. This peer port is used to communicate with local RTI components and with other RTI Assistants. The default port is 6003. You can change the peer port that the RTI Assistant uses.

- To specify the RTI Assistant's port, set the RTI_ASSISTANT_PORT environment variable.

i All RTI Assistants that expect to communicate with each other must use the same port.

Specifying the RTI Assistant's Client Port

Sometimes the RTI Assistant may need to use a different port to communicate with local RTI components than it does to communicate with other RTI Assistants. For example, on Linux machines with multiple users, each user must have their RTI components connect to their own instance of the RTI Assistant. Therefore each user needs to use a different port for their RTI Assistant socket. At the same time, the RTI Assistants must still be able to communicate with each other.

The `RTI_ASSISTANT_CLIENT_PORT` environment variable allows each user to set a port for communication between their local RTI components and the RTI Assistant. RTI Assistants will communicate with each other through the default port or the port specified by `RTI_ASSISTANT_PORT`.

5.2.8 Specifying the RTI Assistant's Multicast Address

RTI Assistants communicate with each other using UDP multicast. The RTI Assistant supports IPv4 and IPv6. You can specify the multicast addresses that RTI Assistants use to communicate with each other. The default address is 229.7.7.10 for IPv4 and ff12::7:10 for IPv6.

- To specify the RTI Assistant's IPv4 multicast address, set the `RTI_ASSISTANT_IPV4_ADDRESS` environment variable.
- To specify the RTI Assistant's IPv6 multicast address, set the `RTI_ASSISTANT_IPV6_ADDRESS` environment variable.

5.2.9 Specifying the RTI Assistant's Multicast Time-to-Live

- To set the RTI Assistant's multicast TTL, create an environment variable called `RTI_ASSISTANT_TTL`. The default value is 1.

5.2.10 Disabling the RTI Assistant

- To disable the RTI Assistant, create an environment variable called `RTI_ASSISTANT_DISABLE`. It does not require a value. Its existence causes the RTI to not create the RTI Assistant.

5.2.11 RTI Assistant Command-Line Options

Table 3 describes the RTI Assistant's command-line options.

Table 3. rtiAssistant command-line options

Option	Description
<code>(--autoExit -K)</code>	Specifies that the RTI Assistant exit automatically after the last connection exits.
<code>(--Connection -C)</code>	Opens the RTI Connections dialog box.

Table 3. rtiAssistant command-line options (continued)

Option	Description
(--disableMulticast -d)	Disables multicast sending and receiving for the RTI Assistant.
 If multicast is disabled on a network, the RTI Assistant network map and federations table may not display properly. This problem does not affect the functioning of the RTI, just the display in the RTI Assistant	
(--Federations -F)	Opens the Federations View.
(--forceShutdown -S)	Shuts down the local RTI Assistant and all connected components without prompting you.
(--History -H)	Opens the Notifications History dialog box.
(--notifyLevel -n) <i>notification_level</i>	Specifies the notification level for application messages, where <i>notification_level</i> is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2.
(--shutdownRequest -s)	Prompts you to shut down the local RTI Assistant and all connected components.
(--ignore_rest --)	Ignores all command-line options that are listed after this argument.
(--version -v)	Displays version information.
(--help -h)	Displays command-line usage information in a console window.

5.3. The RTI Forwarder Application

As explained in "2.3.2 Reliable Transport and TCP Forwarding on a LAN" on page 29, the MAK RTI uses TCP forwarding for all reliable communications. This is done using the RTI Forwarder. When you start the rtiexec, if it is configured to use a local RTI Forwarder, it checks to see if one is running. If an RTI Forwarder is running, the rtiexec connects to it. If an RTI Forwarder is not running, the rtiexec starts one using the same connection configuration that the rtiexec uses.

Since the rtiexec usually requires minimal processing time, federations with smaller reliable message loads can easily be supported by centralized TCP forwarding running on the same machine as the rtiexec. As the reliable message load increases, performance can be significantly improved by running the RTI Forwarder (*rtiForwarder.exe*) on a separate machine.

You may decide to run multiple RTI Forwarders to implement a distributed forwarding scheme. For details, see "Chapter 10. Configuring Message Forwarding" on page 135.

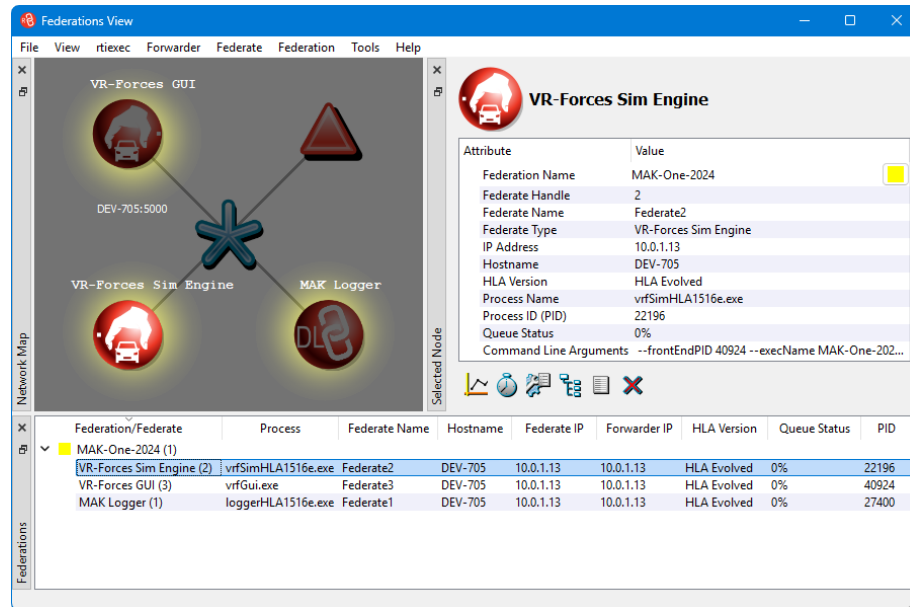
5.3.1 Starting an RTI Forwarder

You can start an RTI Forwarder from the command line and from the Federations View. If a forwarder is not already running, it starts automatically when you start the rtiexec.

► **To start an RTI Forwarder from the Federations View:**

1. In the system tray, right-click the RTI Assistant icon (🔧). A context menu displays.
2. Choose **Federations View**. The Federations View opens (Figure 8).

Figure 8. Federations View



3. Choose **Forwarder > Start Forwarder**, or click the Start Forwarder button (🌟). The Choose Connection Configuration dialog box opens.
4. Select a connection configuration. (If you are creating a distributed forwarder network and have not previously created connection configurations that specify an address for the forwarder that you want to connect to, edit the connection configuration to add the forwarder address. For more information, see "10.6.1 Configuring RTI Forwarders Using the RTI Assistant" on page 146.)
5. In the RTI Settings list, select the RTI Settings configuration you want to use.
6. Click OK.

► **To start an RTI Forwarder from the command line:**

1. Run:

```
./bin/rtiForwarder.exe options
```

Unless you specified the `--manual` command-line option, set **RTI_configureConnectionWithRid** to 1, or configured the RTI to always use a particular connection, the Choose Connection Configuration dialog box opens.

2. Select a connection configuration. (If you are creating a distributed forwarder network and have not previously created connection configurations that specify an address for the forwarder to which you want to connect, edit the connection configuration to add the forwarder address. For more information, see "10.6.1 Configuring RTI Forwarders Using the RTI Assistant" on page 146.)

3. In the RTI Settings list, select the RTI Settings configuration you want to use.
4. Click OK.

Table 4 describes the command-line options for the RTI Forwarder.

Table 4. rtiForwarder command-line options

Option	Description
(--destAddrString -A) <i>address</i>	Specifies the multicast group address to use if reliable transport is disabled. Equivalent to the RTI_destAddrString parameter.
(--distributedForwarderPort -D) <i>port</i>	Specifies the port that RTI Forwarders use to communicate with each other.
(--doNotUseRoutesFile -d)	Disables the use of a routes file for the forwarder.
(--forceFullCompliance -f)	Specifies that the forwarder run in fully compliant mode. This option is ignored unless --manual is used.
--forwarderPortToConnectTo	Specifies a different port for additional forwarders to which you want this forwarder to try to connect. This is an alternative to using a routes file.
(--forwarderToConnectTo -F) <i>address</i>	Specifies the address of an additional forwarder that this forwarder should try to connect to instead of using the routes file. Once connected to the designated forwarder, this forwarder will connect to the forwarders connected to the designated forwarder. As a result, this forwarder will join the fully connected mesh of forwarders.
(--manual -M)	Specifies that the RTI Forwarder use the connection configuration in the RID file or supplied on the command line rather than the configuration stored by the RTI Assistant. It is equivalent to setting RTI_configureConnectionWithRid to 1. Since command-line options override the RID file, using this option enables you to specify the connection configuration from the command line.
(--networkInterfaceAddr -N) <i>address</i>	Specifies the IP address of the network card that you want this forwarder to use.
(--notifyLevel -n) <i>notification_level</i>	Specifies the notification level for application messages, where <i>notification_level</i> is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2.

Table 4. *rtiForwarder* command-line options (continued)

Option	Description
(--routesFileName -r) <i>routeFileName</i>	Specifies the forwarder route file to use. Overrides the RTI_forwarderRoutesFile parameter in the RID file.
(--setLogFileName -l) <i>filename</i>	Specifies the log file name for the forwarder.
(--setRidFileName -R) <i>RID_file</i>	Specifies the RTI initialization data (RTI settings) file to use (for example, <i>rid.mtl</i>). --setRidFileName overrides the RTI_RID_FILE environment variable.
(--setTcpPort -T) <i>port</i>	Specifies the port to use when connecting to the application using TCP. Equivalent to the RTI_tcpPort parameter.
(--setUdpPort -P) <i>port</i>	Specifies the port to listen to. Equivalent to the RTI_udpPort parameter.
(--tcpAddress -t) <i>address</i>	Specifies the IP address used for TCP connections (reliable transport).
(--usePreconfiguredRid -p)	Indicates that the RID file specified by --setRidFileName is the name of a preconfigured RTI Settings instead of a file path.
(--ignore_rest --)	Ignores all command-line options that are listed after this argument.
(--version -v)	Displays version information.
(--help -h)	Displays a list of command-line options.

5.3.2 Shutting Down an RTI Forwarder

► To shut down an RTI Forwarder:

1. In the system tray, right-click the RTI Assistant icon (🔧). A context menu displays.
2. Choose **Federations View**. The Federations View opens (Figure 8).
3. In the Current Connections list, select the connection whose forwarder you want to shut down.
4. Choose **Forwarder > Shut Down Forwarder**, or click the Shut Down Forwarder button (🔴X). A confirmation prompt opens.
5. Click Yes.

5.4. The RTI Federate Protocol Server

The Federate Protocol API relies on the federate connecting to the RTI by making a point-to-point network connection based on TCP/IP. In the MAK RTI, this connection is made with the RTI federate protocol server application (*rtiFedProServer.exe*). The federate protocol server accepts TCP connections using port 15164 by default. You can change it using the --port command-line option.

Once a federate connects, the server expects the federate to send a new session message. The server then creates a federate proxy for the federate protocol session. After the federate proxy is created, the server responds to the federate with a new session status message. The federate can then begin sending service call request messages and expect to receive service callback request messages as appropriate. The first service call request should be the "connect" service call.

5.4.1 Starting the RTI Federate Protocol Server

The RTI federate protocol server can be started independently; however, it is suggested that a conventional RTI configuration (the `rtiexec` and `rtiForwarder`) be initiated before any federate connects with the server. The RTI federate proxies within the federate protocol server will connect and interact with the conventional RTI. It is expected the conventional RTI configuration will have already been established before a "connect" service call is processed. However, in the case of the MAK RTI, the federate proxy can use the RTI Assistant to start up an RTI configuration (or select among RTI configurations). This option often requires manually responding to the RTI Assistant connection dialogs (at least for the first connection).

i The RTI federate protocol server does not require a federate to use the Federate Protocol Wrapper. The wrapper is provided as a convenience.

Table 5 describes the RTI federate protocol server's command-line options.

Table 5. `rtiFedProServer` command-line options

Option	Description
(--address -a) <i>ip_address</i>	The address that the server will listen to for connections.
(--notifyLevel -n) <i>notification_level</i>	Specifies the notification level for application messages, where <i>notification_level</i> is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2.
(--port -p) <i>port_number</i>	The port that the server will listen to for connections. The default is 15164.
(--ignore_rest --)	Ignores all command-line options that are listed after this argument.
(--version -v)	Displays version information.
(--help -h)	Displays command-line usage information in a console window.

5.5. The Federation Shadow Application

The Federation Shadow application (`rtiFederationShadow.exe`) supports the recovery of orphan attributes when federates terminate abnormally (see "6.8.1 Responding to Abnormal Termination" on page 91). The MAK RTI implementation of the ownership management services uses the federate LRCs to manage orphan attributes. When a

federate resigns with orphan attributes under normal circumstances, it transfers them to a designated federate's LRC via the rtiexec. The rtiexec manages which federate is the designated federate (initially the first to join).

In the case of a federate abnormally terminating, there is no opportunity to recover that federate's orphan attributes. To better handle orphan attributes in these abnormal conditions, the Federation Shadow monitors the ownership activity in a federation. It keeps track of the ownership status of all attribute instances. When a federate abnormally terminates, the Federation Shadow steps in for the federate and transfers the federate's orphan attributes to the rtiexec. As in normal operation, the rtiexec then transfers them to the designated federate (which is also captured by the Federation Shadow).

The Federation Shadow is only necessary when a federation is using ownership management and requires the recovery of orphan attributes from federates that terminate abnormally. When enabled through the RTI Settings, the Federation Shadow starts automatically. You can start it manually.

5.5.1 RTI Federation Shadow Command-Line Options

Table 6 describes the RTI Federation Shadow's command-line options.

Table 6. rtiFederationShadow command-line options

Option	Description
(--destAddrString -A) <i>address</i>	Specifies the multicast group address to use if reliable transport is disabled. Equivalent to the RTI_destAddrString parameter. This option is ignored unless --manual is used.
(--enableProfiling -e)	Internal use only.
(--federationHandle -f) <i>federation_handle</i>	Sets the federation handle.
(--federationName -f) <i>federation_name</i>	Sets the name of the federation.
(--manual -M)	Specifies that the RTI Federation Shadow use the connection configuration supplied on the command line rather than the configuration stored by the RTI Assistant.
(--notifyLevel -n) <i>notification_level</i>	Specifies the notification level for application messages, where <i>notification_level</i> is a number from 0 through 4. The least verbose message level is 0, the highest is 4. The default is 2.
(--setLogFileName -l) <i>filename</i>	Specifies the log file name for the rtiFederationShadow.
(--setNotifyQuiet -q)	Suppresses output of diagnostic messages (sets the notification level to 0).
(--setRidFileName -R) <i>RID_file</i>	Specifies the RTI initialization data (RTI settings) file to use (for example, <i>rid.mtl</i>).

Table 6. `rtiFederationShadow` command-line options (continued)

Option	Description
<code>(--setTcpPort -T) port</code>	Specifies the port to use when connecting to the application using TCP. This option is ignored unless <code>--manual</code> is used.
<code>(--setUdpPort -P) port</code>	Specifies the port to listen to. This option is ignored unless <code>--manual</code> is used.
<code>(--tcpAddress -t) address</code>	Specifies the IP address used for TCP connections (reliable transport).
<code>(--tcpNetworkInterfaceAddr -i) address</code>	The IP address of the local network interface used for TCP connections (reliable transport).
<code>(--udpNetworkInterfaceAddr -N) address</code>	Specifies the IP address of the network card that you want this forwarder to use for UDP connections (best effort transport).
<code>(--usePreconfiguredRid -p)</code>	Indicates that the RID file specified by <code>--setRidFileName</code> is the name of a preconfigured RTI Settings instead of a file path.
<code>(--useReliable -r)</code>	Specifies that the RTI Federation Shadow use reliable message transport. It sets RTI_internalMsgReliableWhenUsingRtiexec and RTI_fomDataReliableWhenUsingRtiexec to 1. This option is ignored unless <code>--manual</code> is used.
<code>(--ignore_rest --</code>	Ignores all command-line options that are listed after this argument.
<code>(--version -v)</code>	Displays version information.
<code>(--help -h)</code>	Displays command-line usage information in a console window.

5.6. The RTI Command Line Tool

You can send commands to the RTI from a console window using the RTI command line tool. This tool is not interactive.

► **To use the RTI command line tool:**

1. Open a console window.
2. Change directory to `./bin`.
3. Enter a command in the format:

`rti option`

Table 7 lists the commands that you can use with RTI command line tool.

Table 7. rtiexec runtime commands

Command	Inputs	Description
delete	<i>componentHandle</i> <i>federationHandle</i> <i>federateHandle</i>	Resigns the federate with the given <i>componentHandle</i> , <i>federationHandle</i> , and <i>federateHandle</i> from the rtiexec. Can be used with local and remote federates. To get handles, call: <code>rti list rtiexec</code> The <i>componentHandle</i> is the handle the RTI Assistant uses to designate the different local components connected to it. In the results from the list command, it is the number displayed at the beginning of the line.
list	[all rtiexec rtiForwarder]	Lists components based on the input parameter: • all (or no input). All of the local RTI components are listed. • rtiexec. All federations and federates of the local rtiexecs are listed. • rtiForwarder. All connections of the local RTI Forwarders are listed.
kill	<i>componentHandle</i>	Resigns local federates and shuts down local rtiexecs and RTI Forwarders. To get handles, call: <code>rti list</code> The <i>componentHandle</i> is the number displayed at the beginning of the line.
shutdownall	None.	Shuts down the local RTI Assistant and all connected local RTI components.

This example uses the list command to generate information about components. Then it deletes a federate.

```
C:\source\makRti\bin>rti list rtiexec
Information from local rtiexec Connections
  1) dubois370 (192.168.3.226), 4226, 229.7.7.7, fully compliant
    [192.168.3.226]
  4) MAKsimple
    2) rtisimple1516 (dubois370)
```

```
C:\source\makRti\bin>rti delete 1 4 2
Delete federate Component Command sent to rtiexec
```

This command tells the rtiexec with LRC handle 1, to delete the federate with federate handle 2, in the federation with federation handle 4.

5.6.1 Listing Federation Executions and Federates

To list all running federation executions and their federates, use the `list` command.

Output from the `rti list` command might look like:

```
C:\MAK\makRti3.4\bin>rti list
```

```
Printing Local Components
```

```
Locally Connected rtiexecs by Assistant ID
```

```
0) fw270 (10.0.1.166), 4678, 229.7.7.7, fully compliant  
[10.0.1.166]
```

```
Locally Connected Forwarders by Assistant ID
```

```
1) fw270 (10.0.1.166), 4678, 229.7.7.7, fully compliant  
[10.0.1.166]
```

```
Locally Connected LRCs by Assistant ID
```

```
2) Logger (1)  
3) MAK PVD (2)  
4) VR-Link F18 (3)
```

Output from `rti list rtiexec` would like:

```
C:\MAK\makRti3.4\bin>rti list rtiexec
```

```
Information from local rtiexec Connections
```

```
0) fw270 (10.0.1.166), 4678, 229.7.7.7, fully compliant  
[10.0.1.166]  
2) VR-Link  
1) Logger (fw270)  
2) MAK PVD (fw270)  
3) VR-Link F18 (fw270)
```

5.6.2 Deleting Federates

The `delete` command deletes the specified federate by handle and federation execution name or deletes all federates with the specified handle. You can use `delete` to explicitly resign a federate. This is useful if a federate crashes before it resigns properly. For example, assuming the output from the `list rtiexec` command in "5.6.1 Listing Federation Executions and Federates" above, to delete the Logger federate from the VR-Link federation execution, you would enter the command:

```
kill 1 Logger
```

To delete all federates with the handle 1 from all federation executions, enter the command:

```
kill 1
```



6. Managing Federations

This chapter explains how to view information about and configure federation executions.

6.1. Introduction	84
6.2. Specifying the FED File	84
6.2.1 Search Order for FED Files	84
6.2.2 Distributing the FED File	84
6.3. Using Lightweight Mode	85
6.3.1 Limitations of Lightweight Mode	86
6.4. Viewing Information about a Federation	87
6.4.1 The Network Map	88
6.5. Viewing Time Management State Information	88
6.6. Viewing Synchronization Points	89
6.7. Viewing Federation-Wide Notification Messages	90
6.8. Configuring Fault Tolerance Strategies	91
6.8.1 Responding to Abnormal Termination	91
6.8.2 Reconnecting to the RTI Forwarder	94
6.8.3 Specifying the Response Interval	95
6.8.4 Processing Discovery of Unknown Objects	96
6.8.5 Choosing Silent Failure Instead of Exceptions	96
6.9. Configuring Save/Restore	97

6.1. Introduction

The MAK RTI has a graphical user interface (GUI) for viewing information about federations and federates.

This chapter covers high-level federation management issues and explains how to use the RTI Assistant and RTISpy GUIs to help manage federations.

6.2. Specifying the FED File

When you use the `rtiexec`, the federate that successfully creates the federation execution dictates the name of the FED file for all federates to use by passing it to `createFederationExecution()`. You can configure the RTI so that when a federate joins the federation, the RTI exchanges either just the FED file name or the entire FED file contents between the `rtiexec` and the federate's LRC.

Exchanging just the FED file name is very efficient, but each federate's LRC must read the FED file locally and there may be consistency issues among federates. To prevent problems caused by inconsistent FED files, when the federation is created, the LRC of the creating federate sends the CRC (cyclic redundancy check) of the FED file to the `rtiexec`. Thereafter, as federates join the federation, each LRC sends the CRC of the local FED file. If **RTI_crcCheckFedFile** is set to 0, and the federate's CRC does not match the CRC of the FED file used to create the federation, the joining federate is notified that the FED files are different and it joins the federation. If **RTI_crcCheckFedFile** is set to 1 and the CRCs do not match, the federate cannot join the federation.

Exchanging the entire FED file ensures that each federate uses the same FED file, but introduces some overhead. (For details, see "6.2.2 Distributing the FED File" below.)

 In full compliance mode, **RTI_distributeFedFile** is enabled.

In lightweight mode, the name of the FED file is not communicated among federates. If you want the federates to use a FED file other than the default (the federation execution name with *.fed* appended to the end), then either use the RTI settings parameter **RTI_defaultFedFile** to specify the name of the FED file to read, or at each federate use the `createFederationExecution` service prior to the `joinFederationService` to create a mapping between the federation name and the FED file name.

6.2.1 Search Order for FED Files

Regardless of how the name of the FED file is specified, the MAK RTI looks for the file in the current working directory first. Then it looks in the directory specified by the `RTI_CONFIG` environment variable. If the RTI cannot find the appropriate FED file in either of these locations, the federate cannot join.

6.2.2 Distributing the FED File

You can distribute the FED file from the federate that creates the federation to all joining federates instead of reading a local copy of the file at each federate. When the first federate creates the federation, it passes the FED file to the `rtiexec`. As additional federates join, the `rtiexec` passes the file to them along with their join confirmation.

To enable distribution of the FED file, set the **RTI_distributeFedFile**, **RTI_useRtiExec**, and **RTI_internalMsgReliableWhenUsingRtiexec** (Enable Send Internal RTI Messages Reliable (TCP)) parameters to 1.

 In full compliance mode, **RTI_distributeFedFile** is enabled.

The primary reasons to enable FED file distribution are to ensure that the FOM used by each federate is consistent while you are developing a federation or to support FOM modules. During federation development, the FOM may be in a state of flux. If different federates try to join the federation using different revisions of the FOM, unexpected and difficult-to-diagnose problems may arise.

However, distributing the FED file is a relatively expensive network operation, which may slow the join process and, in the case of late joining federates, may even impede the performance of a running federation. Therefore, unless you are using FOM modules, we recommend that you only use FED file distribution in the federation development phase, and use only statically distributed FED files when the simulation is deployed. (If you use the FOM module feature, you must distribute the FED file. For details, see "13.1. Using FOM Modules" on page 188.)

 Any federate that attempts to create a federation must have a local copy of the FED file, because the createFederationExecution API routine requires a valid FED file.

Distributing an HLA 1516 FDD File

In 1516, the HLA DTD is not distributed with the FDD file. In order for the LRC to process the FDD file, the HLA DTD must be accessible as specified in the FDD DTD attribute. The DTD is found relative to the federate's working directory.

For HLA 1516 Evolved and HLA 4, the DTD file has been replaced with an XSD, which is located on the IEEE website (<https://ieee.org>). This removes the need to distribute the XSD file with the FDD file for HLA 1516 Evolved and HLA 4.

6.3. Using Lightweight Mode

The MAK RTI supports a lightweight mode in which the rtiexec is not necessary. In general, lightweight mode is appropriate for small, simple federations that do not use the more complex RTI services, that require only best effort transport, and that do not require strict compliance with the HLA Specification.

Many MAK RTI users use lightweight mode during development and debugging of their federates because of the convenience of being able to quickly and repeatedly restart federations without worrying about running the rtiexec. However, many switch to using the rtiexec when playing in real exercises so that they can gain strict compliance with the HLA specification, use reliable transport, and take advantage of the fault tolerance and guarantee of federate handle uniqueness that is supported only with the rtiexec.

When you run in lightweight mode, you can use the RTI Assistant to:

- View the notification history.
- Enable federate logging.

- Force a federate to resign.
- Run latency tests.
- Shut down all federates.

 The federation management features of the RTI Assistant are not available in lightweight mode.

6.3.1 Limitations of Lightweight Mode

In lightweight mode, services that require the central processing provided by the rtiexec (reliable transport, time management, save and restore, and synchronization points) are not available. Other RTI services, (including MOM, DDM, and ownership management) work in lightweight mode, but in certain cases functionality is limited. In lightweight mode:

- Released attributes are permanently orphaned. (When a federate resigns from a federation with a resign action `RELEASE_ATTRIBUTES DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES`, the RTI is supposed to allow other federates to acquire ownership of attribute instances that the resigning federate is releasing.)
- There is no central place to store information about which federation executions exist and which federates are joined to them. Therefore:
 - The `createFederationExecution()` service returns successfully even if the named execution has already been created.
 - The `destroyFederationExecution()` service always returns successfully even if the named execution does not currently exist.
 - The `joinFederationExecution()` service returns successfully even if the named federation execution has never been created using `createFederationExecution()`.
- Each federate unilaterally chooses its federate ID without asking the rtiexec to make sure it is unique. This makes it possible for two federates to have the same federate ID. (If this occurs, you will need to restart your federation.) The chances of two federates choosing the same ID depend on the strategy the federate uses for choosing its ID. By default, each federate uses its process ID modulo 10,000 as its federate ID. But if **RTI_useRandomNumberForFedHandle** is set to 1 in your RTI settings, each federate chooses a handle at random.
- The first three characters plus the last two characters of the federation execution name are used to construct a federation execution identifier. If you have more than one federation execution running at the same time on the same port or address, their names must differ within the first four characters or the last character.
- The RTI cannot detect crashed federates and automatically resign them from the federation execution.
- The RTI does not pass the name of the FED file from the federate that created the federation execution to the other federates. If your FED file name is something other than the name of your federation execution followed by *.fed*, you must call `createFederationExecution()` in every federate, passing the name of the appropriate

FED file.

- When using RTI 1516 or HLA Evolved, name reservation is only checked against what is known at the local RTI component (previously reserved names, registered objects, and discovered objects).
- FOM module merging is not supported.

6.4. Viewing Information about a Federation

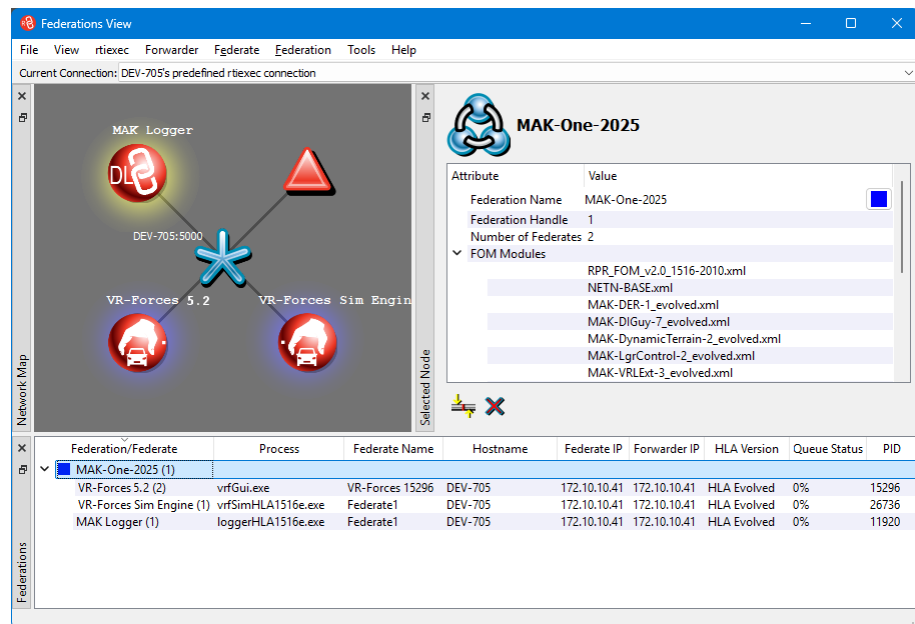
The Federations View displays information about federations. It has the following panels:

- Network Map. The Network Map display icons for the RTI Forwarder, the rtiexec, and the federates in the selected connection.
- Selected Node. The Selected Node panel displays details about the node that is selected in the Network Map.
- Federations. The Federations panel lists the federations that are active on the selected connection and the federates in each federation.

► To display information about a federation in the Federations View:

1. In the system tray, right-click the RTI Assistant icon (🔧). A context menu displays.
2. Choose Federations View. The Federations View opens (Figure 9).
3. In the Current Connection box, select the connection that you want to view.

Figure 9. RTI Federation View, federation selected



4. Select a federation in the Federations panel. Information about the federation displays in the Selected Node panel.

6.4.1 The Network Map

The Network Map (Figure 9 on the previous page) displays an icon for each forwarder, each federate, and the rtiexec node. The links between components represent the reliable connections (TCP) between them. (Components can communicate through other connectionless methods that are not represented in the network map (UDP).)

When you select a node on the map, details about the node display in the Selected Node panel.

 To view the network map, you must run the rtiexec and must run internal message reliable.

6.5. Viewing Time Management State Information

When time management is enabled, the Federations View can display time management state information for federates. You can see the following information:

- Regulating – Is the federate regulating?
- Constrained – Is the federate constrained?
- LBTS – The Lower Bound Time Stamp on messages that the federate may generate. This view of LBTS is the time to which a regulating federate constrains all other constrained federates, rather than the minimum LBTS by which the federate is itself constrained. When the federates are sorted by the LBTS value, the minimum LBTS value indicates the LBTS of messages that any federate may receive (except the smallest federate that uses the next smallest value).
- Lookahead – The federate's current lookahead setting.
- Current Time – The federate's current view of simulation time. This is the time of the federate's last Time Advance Grant, and is unaffected when the federate enters the Time Advancing state.

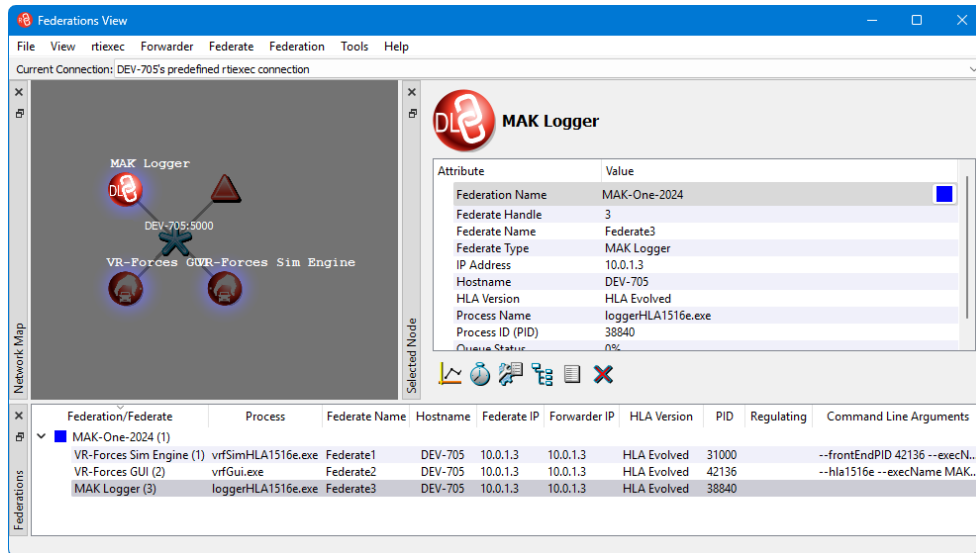
-  • In the Federations View, you can configure which time management state information displays.
- Displaying time management information adds some overhead to RTI performance.

► **To display time management state information:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. In the Current Connection box, select the connection that you want to view.
4. Choose **View > Federate Columns > Customize**.
5. Select the time management columns that you want to display.
6. Click OK. The columns are added to the Federations panel (Figure 10).

i You can also display or hide columns individually by selecting them on the Federate Columns sub-menu.

Figure 10. Time management state information



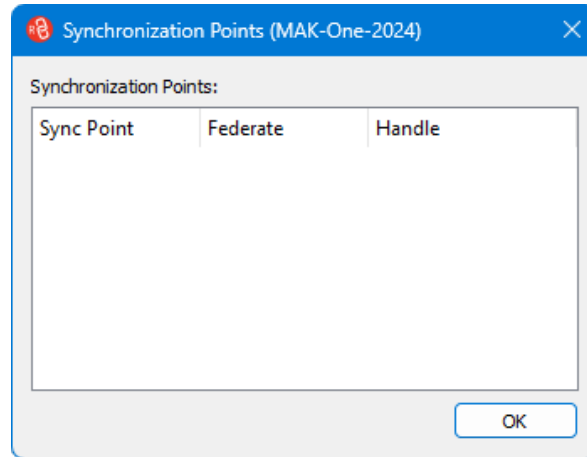
6.6. Viewing Synchronization Points

You can view the status of a federation's synchronization points in the Federations View and RTIs Spy.

i The Federations View displays only pending synchronization points. Once the point has been reached, the information is deleted from the window.

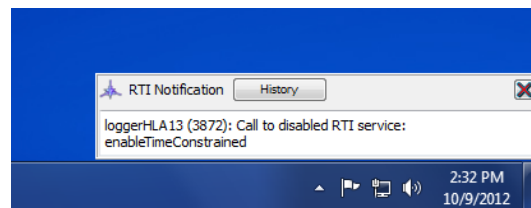
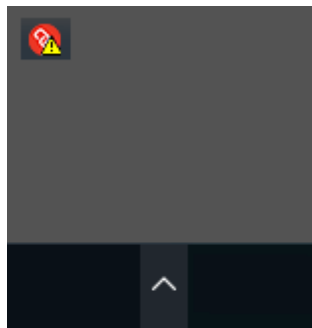
► To view synchronization points:

1. In the system tray, right-click the RTI Assistant icon (RTI Assistant icon). A context menu displays.
2. Choose Federations View. The Federations View opens (Figure 9).
3. In the Current Connection box, select the connection that you want to view.
4. In the Federations panel, select the federation whose synchronization points you want to view.
5. Choose **Federation > Synchronization Points** or click the Synchronization Points button (Synchronization Points button). The Synchronization Points dialog box opens (Figure 11). It lists pending synchronization points.

Figure 11. Synchronization points

6.7. Viewing Federation-Wide Notification Messages

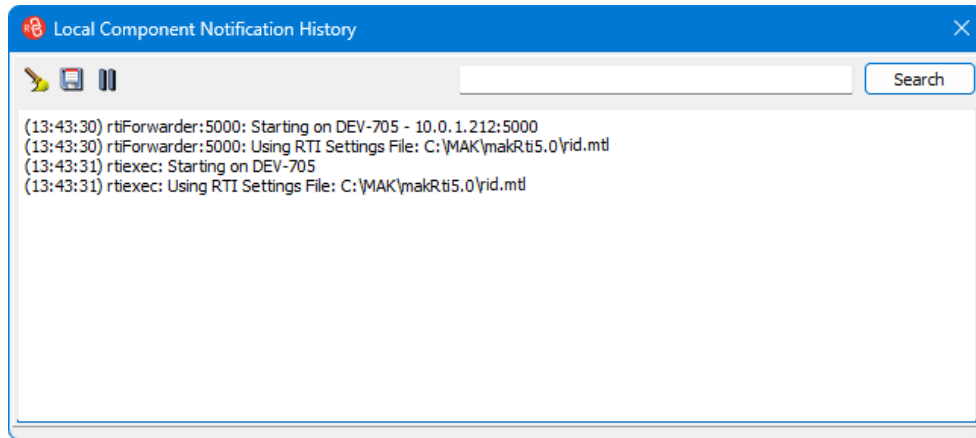
The RTI Assistant displays Fatal and Warn messages received from all federates running on the same machine. The messages are displayed in a popup window when they are received (Figure 12). Thereafter, you can view them in the Local Component Notification History window. An exclamation point is added to the RTI Assistant icon until the next time the notification history is viewed (Figure 13). You can clear, pause, and save the notification history. (For details, see "5.2.6 Common Operations in RTI Assistant Log Windows" on page 71.)

Figure 12. RTI notification popup window**Figure 13. RTI Assistant icon message prompt**

► **To display the RTI Notification History:**

1. In the system tray, right-click the RTI Assistant icon (🔊).
2. On the context menu, choose Local Notification History. The Local Component Notification History window opens (Figure 14).

Figure 14. Local Component Notification History window



6.8. Configuring Fault Tolerance Strategies

Fault tolerance is a set of strategies for responding to problems within a federation, such as a federate dropping out without resigning or unusual delays in connecting to the RTI.

6.8.1 Responding to Abnormal Termination

The MAK RTI supports automatic removal of abnormally terminated federates and their orphaned objects. The RTI detects terminated federates:

- It detects broken TCP connections between the LRC and the rtiexec (that is, by its RTI Forwarder component).
- It monitors a heartbeat message sent by each LRC to the federation execution.

To detect broken TCP connections, an LRC must establish a TCP connection to the rtiexec's RTI Forwarder (**RTI_tcpForwarderAddr**). When the rtiexec detects that this TCP connection has been lost, the federate is forcibly removed from its federation, if it is joined.

The heartbeat mechanism has each LRC send a heartbeat message to its federation executive once it has joined. If the federation executive does not receive a heartbeat message within a specified time-out interval, the federate is forcibly removed from the federation. To configure the heartbeat interval, enable the Enable Federate Heartbeat, and set the Heartbeat Interval. The default heartbeat is 10 seconds.

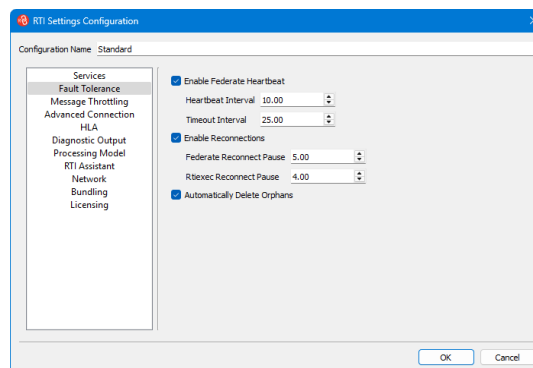
To configure the timeout interval, enable the Enable Federate Heartbeat RTI setting, and set the Timeout Interval setting. The default is 25 seconds.

i The asynchronous IO thread in the asynchronous IO or asynchronous message processing modes does not generate heartbeats. Heartbeats are generated during the call to `tick()`, `evokeCallback()`, or `evokeMultipleCallbacks()`. Therefore the federate must call these services more frequently than the timeout interval.

► **To edit the Enable Federate Heartbeat, Heartbeat Interval, and Timeout Interval RTI settings:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 2). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Fault Tolerance page (Figure 15).

Figure 15. Fault Tolerance page



8. Select the Enable Federate Heartbeat check box.
9. Enter a value in the Heartbeat Interval box.
10. Enter a value in the Timeout Interval box.
11. Click OK.

Handling Orphaned Attributes

Attributes in HLA can become orphaned in situations where no federate owns that attribute instance. This can happen when registering an object without publishing all attributes of that object, when unconditionally divesting attributes, or when a federate resigns with an unconditional divest directive.

Federation Shadow

Under normal conditions, these orphaned attributes can be recovered through HLA ownership transfer services. In cases of abnormal terminations, an RTI component called the Federation Shadow is required for recovering ownership of orphaned attributes. When the Federation Shadow is not enabled and a federate abnormally terminates, orphaned attributes can enter a state where ownership cannot be recovered. The Federation Shadow is disabled by default and should only be enabled in cases where orphaned attributes are possible and must be recoverable. For example, a federation uses ownership management services while federates are set to an automatic resign directive that includes unconditional divestment.

► To enable or disable the Federation Shadow:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 18 on page 98).
8. Select or clear the Enable Federation Shadow check box.
9. Click OK.

Automatic Deletion of Orphaned Attributes

The Automatically Delete Orphans RTI setting provides an alternative way for the RTI to handle orphaned attributes in cases where recovering orphaned attributes is not required. If Automatically Delete Orphans is enabled and the federate owns the `privilegeToDelete` attribute, the orphaned attributes are deleted from the federation. If the federate does not own the object's `privilegeToDelete` attribute, the federate's owned attributes become orphaned and they cannot be recovered. If Automatically Delete Orphans is disabled, the federate's owned attributes become orphaned even when the federate owns the `privilegeToDelete` attribute. A federation can choose to allow orphaned attributes so that attributes owned by other federates can still be updated.

 For HLA 1.3 and HLA 1516 federates, the Automatic Delete Orphans setting sets a resign action of Delete Objects when enabled and No Action when disabled. For HLA Evolved and HLA 4, a federate that is abnormally terminated is resigned with the action specified by the `automaticResignAction` in the FDD file, which takes precedence over the Automatic Delete Orphans setting.

► **To specify deletion of orphaned objects:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon () . A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button () . The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Fault Tolerance page (Figure 15 on page 92).
8. Select or clear the Automatically Delete Orphans check box.
9. Click OK.

6.8.2 Reconnecting to the RTI Forwarder

You can configure an LRC to try to re-establish its connection to the RTI Forwarder if the connection breaks. This feature may allow the federate to continue running despite a large class of network layer errors.

To reconnect to the RTI Forwarder, set **RTI_reconnectEnabled** to 1 (Enable Reconnections), and specify a value for **RTI_federateReconnectPause** (Federate Reconnect Pause) and **RTI_rtiExecReconnectPause**. If reconnect is enabled, when a broken connection is detected, the federate waits for the time specified by **RTI_federateReconnectPause** to allow the network to stabilize, then it tries to re-establish the broken connection. If an rtiexec detects a broken connection to the forwarder, the rtiexec waits for the time specified by **RTI_rtiExecReconnectPause** before it tries to re-establish the connection. The value for **RTI_rtiExecReconnectPause** must be lower than the value of **RTI_federateReconnectPause**. If the rtiexec and some number of federates are disconnected from the forwarder, it is important that the rtiexec re-establish its connection before any federates try to reconnect; otherwise the federates will not succeed. A connection will be dropped if the rtiexec or federate does not re-establish it within the time: $(2 \times \text{ReconnectPause}) + \text{response_interval}$. (For details about specifying the response interval, see "6.8.3 Specifying the Response Interval" on the facing page.)

If the federate reconnects, it usually continues to operate in the federation with no visible effects to the application. If the federate cannot reconnect, the LRC throws an `RTIInternalError` exception to indicate that the federate has lost its connection to the federation (as it does when **RTI_reconnectEnabled** is disabled). During the reconnection period some reliable messages may be lost, because the federate is temporarily disconnected from the RTI.

► **To edit the Enable Reconnections and Federate Reconnect Pause RTI setting:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon () . A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button () . The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Fault Tolerance Page (Figure 15 on page 92).
8. Select the Enable Reconnections check box.
9. Enter a value in the Federate Reconnect Pause box.
10. Click OK.

6.8.3 Specifying the Response Interval

You can control the amount of time (in seconds) that an LRC waits to receive a response from the rtiexec when processing the `createFederation()`, `destroyFederation()`, or `joinFederation()` service calls. You set the response interval with the Response Interval RTI setting. The default interval is 3.0 seconds.

A federate's LRC can sometimes experience unusually long delays when communicating with the rtiexec to process the `createFederation()` or `joinFederation()` service calls. As a result, the federate will not be able to complete its connection to the rtiexec and will report, "Could not connect to rtiexec". The service call throws an `RTIInternalError` exception. The solution to this problem is to increase the amount of time that the LRC waits for a response.

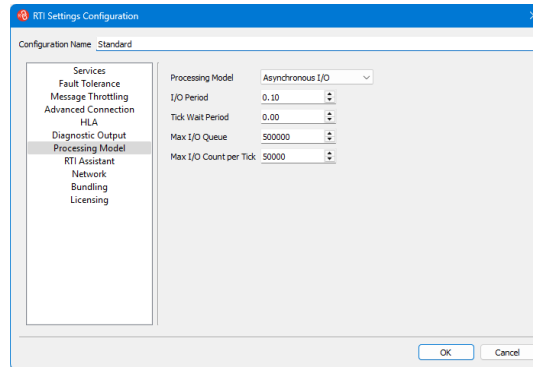
 Other connection issues may cause the LRC to report the same type of connection error, for example, mismatched UDP ports.

► **To edit the Response Interval RTI setting:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon () . A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.

6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Processing Model page (Figure 16).

Figure 16. Processing Model page



8. Enter a value in the Response Interval box.
9. Click OK.

6.8.4 Processing Discovery of Unknown Objects

By default, the RTI does not perform discovery processing for updates of unknown objects. The RTI setting **RTI_processUnknownUpdatesForDiscovery** allows you to control whether updates from unknown objects can cause discovery. By default this option is turned off to improve efficiency. Turning it on provides fault tolerance for dropped register messages when internal messages are sent using UDP. The syntax is:

```
(setqb RTI_processUnknownUpdatesForDiscovery mode)
```

where *mode* is:

- 0 – do not process unknown updates for discovery.
- 1 – process unknown updates for discovery.

Default: 1.

⚠ **RTI_processUnknownUpdatesForDiscovery** should almost always be disabled when **RTI_internalMsgReliableWhenUsingRtiexec** is enabled. **RTI_processUnknownUpdatesForDiscovery** was designed specifically to mitigate problems encountered when reliable messaging is disabled. So, in almost all cases, this setting provides no benefit when **RTI_internalMsgReliableWhenUsingRtiexec** is enabled. Enabling both parameters may lead to unexpected results in some configurations. For example, an object may have an update delivered after it has been deleted, resulting in its mistaken rediscovery.

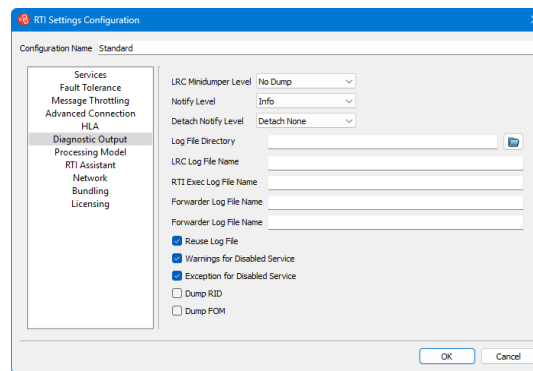
6.8.5 Choosing Silent Failure Instead of Exceptions

You can have your function calls fail silently rather than throwing an exception when they call a service that is not activated in the MAK RTI. Use the Exception for Disable Service RTI setting.

► **To edit the Exception for Disabled Service RTI setting:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Diagnostic Output page (Figure 17).

Figure 17. Diagnostic Output page



8. Select the Exception for Disable Service check box.
9. Click OK.

6.9. Configuring Save/Restore

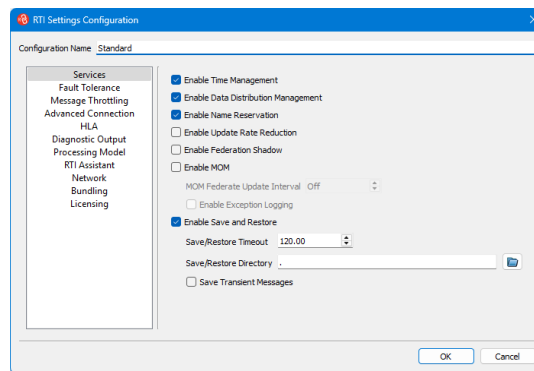
You can configure the number of seconds the federation execution waits for a save/restore operation to complete before indicating a failure with the Save/Restore Timeout RTI setting. The default is 120 seconds.

The heartbeat mechanism is not necessary when reliable transport is enabled and a TCP connection is established by the federate. The detection of a broken TCP connection is sufficient and better than the heartbeat. However, a broken TCP connection is often not detected until the federate sends a message. If a federate is often in a quiescent state in which it is not sending any messages, enabling the heartbeat is one way to ensure early detection of the broken TCP connection. As long as the federate is ticking the RTI, it sends the heartbeats automatically.

► **To enable or disable save/restore:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 18).

Figure 18. Services page



8. Select the Enable Save and Restore check box.
9. Optionally, change the Save/Restore Timeout value.
10. Optionally, specify a location for the Save/Restore Directory.
11. Click OK.



7. Managing Federates

This chapter explains how to view information about federates and how to configure them.

7.1. Displaying Federate Information	100
7.2. Displaying FOM Object and Interaction Classes	100
7.3. Viewing LRC Diagnostic Messages	101
7.3.1 Enabling LRC Diagnostic Data Logging	102
7.3.2 Displaying the Federate Log	102
7.4. Configuring Federate Connections in the RTI Settings	103

7.1. Displaying Federate Information

The RTI Assistant Federations View and the RTIsPy can display the following information about a federate:

- Federate identification information.
 - Federate object and interaction information.
 - Configuration settings (RID (RTI initialization data) file).
 - FOM objects and interactions.
 - Federate log file.
- To display federate information, select a node in the Network Map or a federate listing in the Federations window. Details are displayed in the Selected Node panel.

7.2. Displaying FOM Object and Interaction Classes

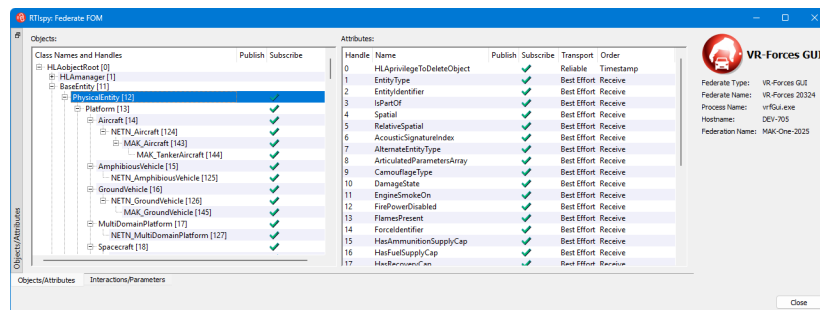
The RTIsPy lets you view a list of the FOM's objects and interactions. It shows which objects and interactions it is publishing and subscribing to.

i If the Network Statistics Monitoring RTI setting is disabled, the RTIsPy: Federate FOM window does not work. For details, see "8.3.1 Enabling Federate Statistics Monitoring" on page 112.

► To display FOM object classes:

1. In the Federations View, select a federate.
2. Choose **Federate > RTIsPy: Federate FOM**, or click the RTIsPy: Federate FOM button () on the Federations View toolbar. The RTIsPy: Federate FOM window opens.
3. Select the Objects/Attributes tab (Figure 19).

Figure 19. RTIsPy: Federate FOM, object classes

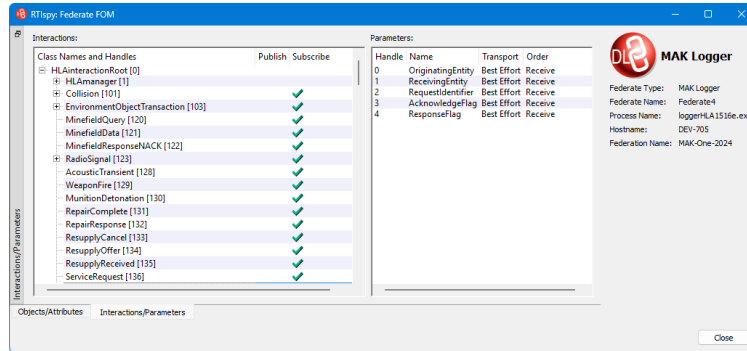


4. Select a class in the Objects list (left pane) to see its attributes displayed in the Attributes list (right pane).

► To display FOM interaction classes:

1. With a federate selected in the Federations View, choose **Federate > RTIsPy: Federate FOM**, or click the RTIsPy: Federate FOM button () on the Federations View toolbar. The RTIsPy: Federate FOM window opens.
2. Select the Interactions/Parameters tab (Figure 20).

Figure 20. RTIsPy: Federate FOM, interaction classes



3. Select a class in the Interactions list (left pane) to see its attributes displayed in the Parameters list (right pane).

7.3. Viewing LRC Diagnostic Messages

The RTI can capture diagnostic messages sent by LRCs. You have two options for enabling message logging:

- Enable logging in the RTI settings file. If you enable logging, all messages are logged to a specified file from the time the federate starts until it exits. You can view these messages by opening the log file in a text editor. You can also view the log at any time while the federate is running and save the current snapshot of messages to a file.

 When logging is enabled in the RTI settings file, if there is more than one federate running on a PC, a separate log file of the specified name is created in the directory in which the executable is located. If you are running more than one executable from the same directory, for example the VR-Link talk and listen applications, data for both federates is written to the same log file unless you disable the Reuse Log File RTI setting.

- Enable logging from the RTI Assistant. If you enable logging from the RTI Assistant, no messages are logged until you enable logging, so startup messages are not captured. The messages are not saved unless you explicitly save them. You can view the log at any time while the federate is running and save the current snapshot of messages to a file. Any messages that are sent between the time you save a snapshot and shut down the federate are not saved.



- Once you start logging messages, either globally or for a particular LRC, you cannot turn off the logging process. However, you can change the notification level to reduce the number of messages logged.
- You can view log files only for federates on the local machine.

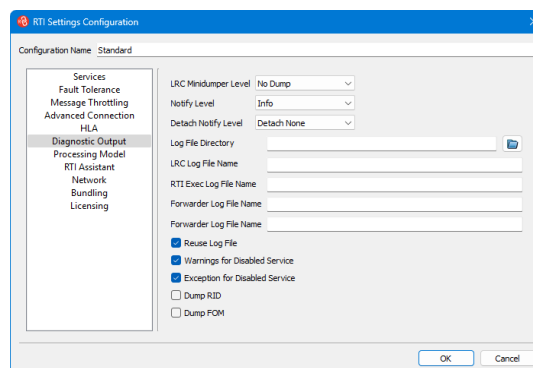
7.3.1 Enabling LRC Diagnostic Data Logging

The Log File Name RTI setting enables or disables logging of LRC diagnostic data. It is disabled by default to improve performance. To enable logging, specify a name for the log file. To disable logging, leave the value blank.

► **To edit the Log File Name RTI setting:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Diagnostic Output page (Figure 21).

Figure 21. Diagnostic Output page

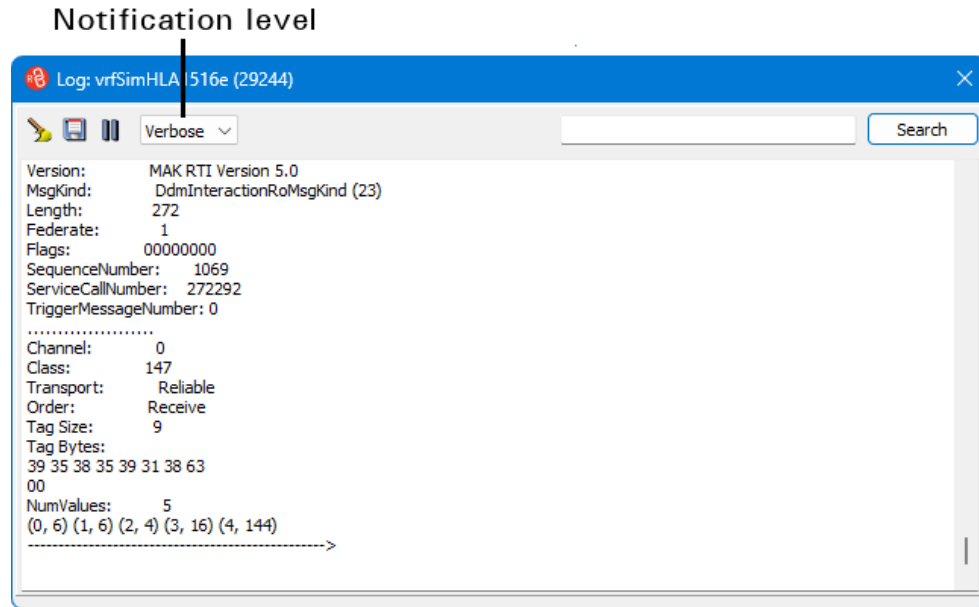


8. Enter a filename in the appropriate Log File Name box.
9. Click OK.

7.3.2 Displaying the Federate Log

The Federate Log window (Figure 22) displays messages based on the notification level in the window. You can pause the display of messages, save them to a file, and clear the window of all messages. (For a description of common operations in log file windows, see "5.2.6 Common Operations in RTI Assistant Log Windows" on page 71.)

Figure 22. Federate Log window



► To display the message log for a federate:

1. In the Federations View, select a federate.
2. Choose **Federate > Federate Log File**, or click the Federate Log File button (📄). If logging is not enabled, you are prompted to enable it.
3. If you are prompted to enable logging, click Yes. The Log window opens.

7.4. Configuring Federate Connections in the RTI Settings

By default, federates connect to a federation by specifying a connection configuration in the RTI Assistant. To configure connection settings in RTI settings, instead of using the RTI Assistant, specify values for these parameters:

- To configure a federate to try to connect to the rtiexec, set **RTI_useRtiExec** to 1.
- To configure federates for lightweight mode, set **RTI_useRtiExec** to 0.
- To specify full compliance mode, set **RTI_forceFullCompliance** to 1.
- Specify the TCP port with **RTI_tcpPort**.
- Specify the UDP port with **RTI_udpPort**.
- Specify the best effort address with **RTI_destAddrString**.
- Specify the RTI Forwarder address with **RTI_tcpForwarderAddr**.
- Set the **RTI_internalMsgReliableWhenUsingRtiexec**, **RTI_fomDataTransportTypeControl**, and **RTI_mcastDiscoveryEnabled** parameters to your preference. (You can also set the first two parameters in the RTI Settings page of the RTI Assistant Preferences: Send Internal RTI Messages Reliable (TCP), FOM Data Transport Type.)
- To force the RTI to use the values specified for these parameters instead of those in the connection configuration, set **RTI_configureConnectionWithRid** to 1.



8. Monitoring the Network

This chapter explains how to monitor network components and the federates in a federation.

8.1. Introduction	106
8.2. Testing Latency Between Federates	106
8.2.1 Enabling Latency Testing	107
8.2.2 Running Latency Tests	107
8.2.3 Interpreting the Results of Latency Tests	108
8.2.4 Running Latency Tests Manually	110
8.2.5 Running Continuous Latency Tests	111
8.2.6 Configuring Latency Tests	111
8.3. The RTIspy Network Monitoring Tool	112
8.3.1 Enabling Federate Statistics Monitoring	112
8.3.2 Displaying Object Statistics	113
8.3.3 Displaying Interaction Statistics	114
8.3.4 Displaying RTI Message Statistics	115
8.3.5 Displaying RTI Services Statistics	115
8.3.6 Displaying Callback Statistics	116
8.3.7 Graphing Federate Statistics	116
8.4. Examples of Using the RTIspy Network Monitoring Tool	118
8.4.1 Finding Bottlenecks Caused by Processing Callbacks	118
8.4.2 Identifying the Cause of Slowdowns Under Time Management	118
8.4.3 Comparing the Costs and Benefits of RTI Features	119

8.1. Introduction

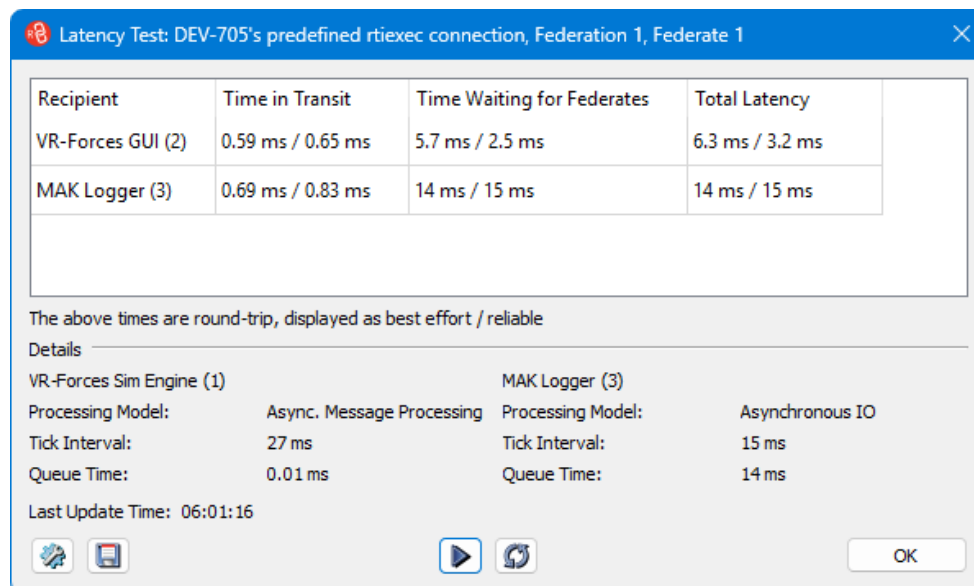
The HLA API provides many options for optimizing performance. Since each of the RTI options offers trade-offs between CPU usage, network overhead, and functionality, it can be very difficult to predict the final effect of the various options on overall network and simulation performance. For example, time management provides synchronization between federates, but imposes additional overhead in terms of message processing and increased complexity at the RTI level. Choosing data distribution management (DDM) over declaration management can greatly reduce the number of updates and interactions sent by the RTI, but adds processing overhead to each RTI call, which may result in an update or interaction message being sent, and in fact, may increase the number of messages sent under certain conditions.

To help you assess the results of using different performance optimizations, you can enable and run latency tests between federates and you can use the RTIs' network monitoring tools to view object and interactions statistics, the frequency of RTI message types, and API call statistics.

8.2. Testing Latency Between Federates

The MAK RTI can run latency tests among federates. You must have at least two federates in the federation to run latency tests. When you start latency testing for a federate, the RTI automatically runs a latency test from this initiating federate to all other federates in the federation. The results are displayed in the Latency Test window (Figure 23). If you select a federate in the Latency Test window, information about the federate displays in the Details panel.

Figure 23. Latency Test window



i The rtiexec must be running to do latency tests.

You can run additional tests manually or run tests continually.

You can configure:

- The number of interactions run in each test.
- The time between interactions.
- The size of the interaction message.

Each test does the following for each interaction that it sends:

1. Records the start time.
2. Sends an interaction from the initiating federate to a recipient federate. The recipient returns the interaction.
3. Waits until the amount of time calculated as `start_time + time_between_interactions` has elapsed.

While the initiator is sending interactions, it is also processing the return interactions and recording individual results. After the last interaction is sent to a given recipient federate and the wait time has expired (plus some extra time), a complete result for the initiator:recipient pair is computed and displayed.

8.2.1 Enabling Latency Testing

Latency tests are run between federates. Latency testing is enabled at the federation level. You can enable and disable latency testing in the Federations View.

► **To enable or disable latency testing in the Federations View:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose **Federations View**. The Federations View opens.
3. In the Current Connection box, select the connection that you want to view.
4. In the list of federations, select the federation that you want to test.
5. Choose **Federation > Latency Testing Enabled**. A checkmark next to the menu command indicates that testing is enabled.

8.2.2 Running Latency Tests

You can run latency testing when the rtiexec is running and in lightweight mode. You must enable latency testing before you can run a test.

 You must have at least two federates in the federation to run latency tests.

► **To run a latency test:**

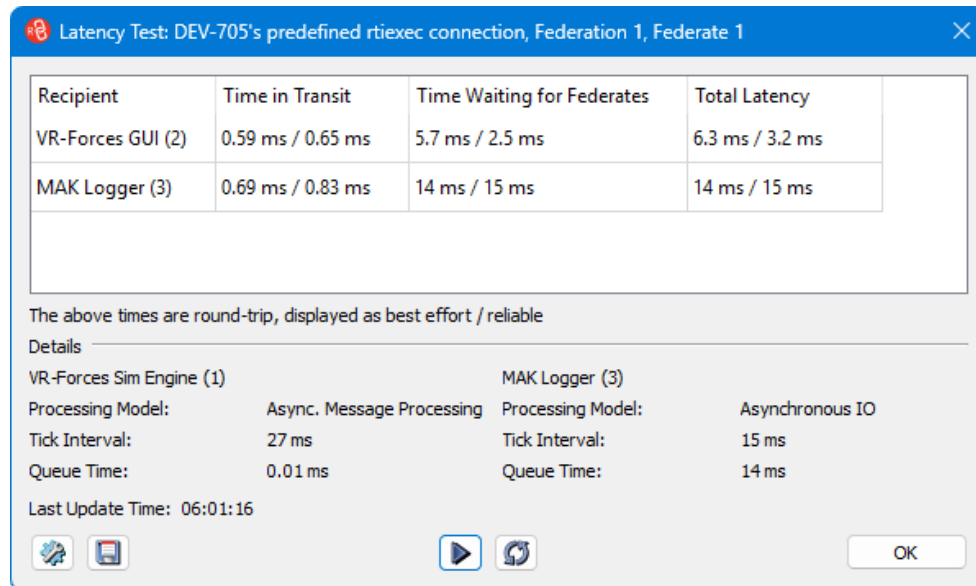
1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose **Federations View**. The Federations View opens.
3. In the Current Connection box, select the connection that you want to view.
4. If latency testing is not enabled, choose **Federation > Latency Testing Enabled**.

5. Select the federate that you want to test.
6. Choose **Federate > Run Latency Test**. The Latency Test window opens (Figure 23 on page 106). The federate name is shown in the title bar.

8.2.3 Interpreting the Results of Latency Tests

The Latency Test window displays the results for tests between the initiating federate and each recipient in a table in the results window. It displays additional details in the Details group box. The Latency Test window displays results for best effort and reliable mode latency (Figure 24). If results are not available for a particular transport type, a dash displays.

Figure 24. Latency test results, best effort only



The data reported in the latency test table is:

- **Time in Transit.** The time in transit is the total latency of the message's round trip minus the time spent in queue waiting to be read, if any. The comparison of time in transit to total latency can help you understand the effect of using different processing modes and tick rates in your federates. For details, see "Understanding the Time in Transit Latency Test Results" on the facing page.
- **Time Waiting for Federates.** The time waiting for federates is the amount of time a message spends in a queue waiting for a federate to retrieve it. In other words, it is the difference between the time in transit and total latency.
- **Total Latency.** Total latency is the round trip time – the time from when the initiator sends the interaction to when the recipient's response is received.

i Each of the values reported is the average of the values for the individual interactions sent in the test.

Understanding the Time in Transit Latency Test Results

The total latency of a message is the sum of the transit time of the message and any time that it has spent in a queue waiting for a federate to read it. The transit time is a function of network topology and hardware. The time spent in queue depends on the queuing strategy of the RTI and how frequently a federate ticks the network for new messages. Therefore, the time in transit values in the Latency Test window must be evaluated in the context of the processing model being used. (The MAK RTI's processing models are described in "2.2. The MAK RTI Process Model" on page 26.) The process models are:

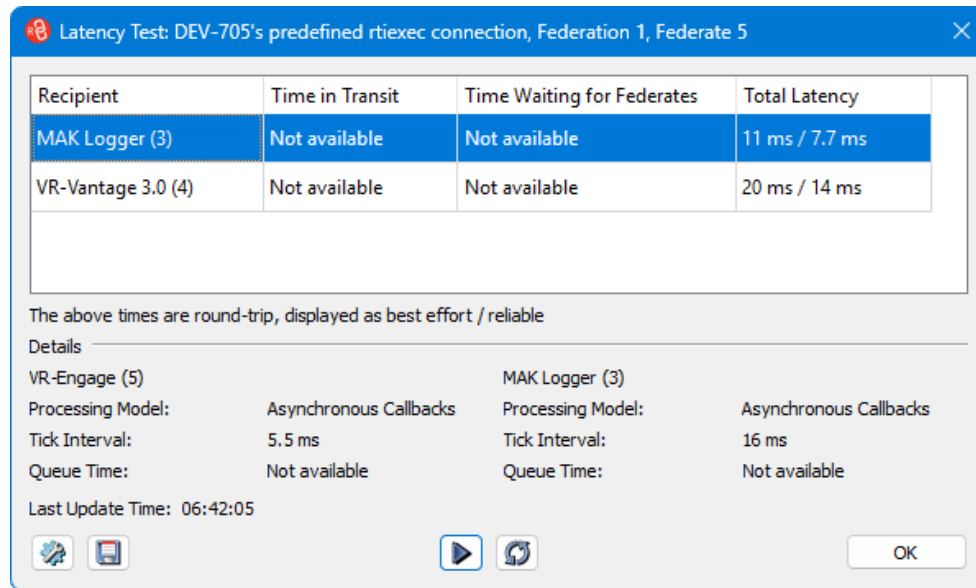
- Synchronous
- Asynchronous
- Asynchronous message processing
- Asynchronous callbacks

In synchronous mode, and asynchronous callbacks mode, there is no internal RTI queue. Therefore, there should be no difference between the time in transit and total latency time. The latency test does not report time in transit or time waiting for federates (Figure 25 on the next page).

The asynchronous and asynchronous message processing modes use message queues to store messages until a federate ticks the RTI and retrieves the message for processing. The total latency time does not tell you how much time a message has been in transit and how much time it has spent sitting in a queue. If you want to improve the performance of your federation, you need to separate these issues so that you know where to focus your efforts. Therefore, during latency testing, the RTI records the time when a message is placed on a queue and when it is removed so that it can report transit time and waiting time separately.

In asynchronous mode, if there is a significant discrepancy between your federate's time in transit and total latency (as shown by the time waiting for federates value), increasing the frequency with which your federate ticks the RTI may improve performance.

In asynchronous message processing mode, the RTI does some message processing before it queues the message. As a result of this preprocessing, when the federate ticks the RTI and retrieves the message it has less work to do. You can compare the latency results of asynchronous mode to asynchronous message mode to see if your federates improve latency by changing the processing model. Increasing the tick rate may still provide improved performance in this mode.

Figure 25. Latency test, asynchronous callbacks mode

Displaying Latency Test Details for a Federate

The Details section of the Latency Test window (Figure 25 above) displays:

- The timestamp of when the report was received and displayed.
 - The federate name (handle) of the initiator and recipient.
 - The processing model of the initiator and recipient, which can be:
 - Synchronous
 - Asynchronous
 - Asynchronous message processing
 - Asynchronous callbacks.
 - The average frequency with which each federate calls tick().
 - The average amount of time that a message was queued at each federate.
- To display details about a federate, select it in the Latency Test window. The Details group box updates.

8.2.4 Running Latency Tests Manually

After the RTI automatically runs its first latency test, you can run additional tests manually.

The buttons in the window are:

-  – Configure test preferences
-  – Run
-  – Run test continuously

- To run a latency test manually, in the Latency Test window, click the Run button (▶).

8.2.5 Running Continuous Latency Tests

You can have the RTI run continuous latency tests. There is no configurable break between each test run. The results of each test replace the results of the previous test in the Latency Test window.

i If a federate drops out of an exercise, its last test results remain in the Latency Test window. They do not update, but there is no indication that the federate is no longer active.

- To run latency tests continuously, in the Latency Test window, click the Run Test Continuously icon (🔄).

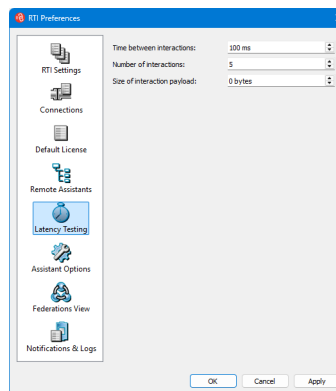
8.2.6 Configuring Latency Tests

You can configure latency test parameters in the Latency Test window or in the RTI Preferences dialog box.

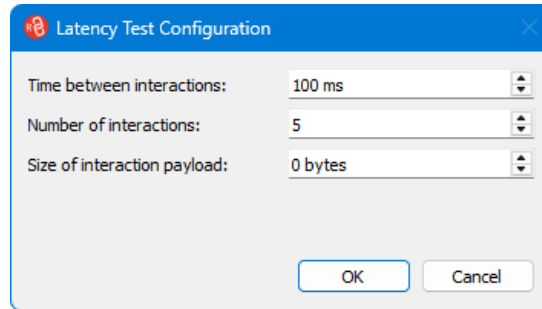
- **To configure latency test parameters:**

1. Display latency test configuration options in one of the following ways:
 - On the RTI Assistant context menu, choose **Preferences**, then select the Latency Testing page (Figure 26).

Figure 26. RTI Preferences dialog box, Latency Testing page



- In the Latency Test window, click the Configure Test Preferences icon (⚙️). The Latency Test Configuration dialog box opens (Figure 27).

Figure 27. Latency Test Configuration dialog box

2. Edit the as needed:
 - Time between interactions. The amount of time, in milliseconds, to wait between sending test interactions.
 - Number of interactions. The number of interactions to send per latency test.
 - Size of interaction payload. The size, in bytes, of the interaction message to send.

8.3. The RTIsPy Network Monitoring Tool

The RTIsPy Network Monitoring Tool is a window into the "black box" of the RTI. It records each message sent and each API call made. You can use this information to help weigh the costs and benefits of different RTI features. It can also help you locate the bottlenecks in your simulations, whether they are due to the underlying network, processing in the RTI, processing in Federate Ambassador callbacks, or the simulation execution itself.

The RTIsPy Network Monitoring Tool exposes the internal processing of the RTI:

- The Callbacks page can show you which RTI Ambassador or Federate Ambassador calls might be affecting execution.
- You can observe how much data is sent across the network to represent the simulation's objects and interactions. This can help identify possible problems and optimizations.
- You can log the statistics to a file, allowing deeper analysis after a simulation run.

8.3.1 Enabling Federate Statistics Monitoring

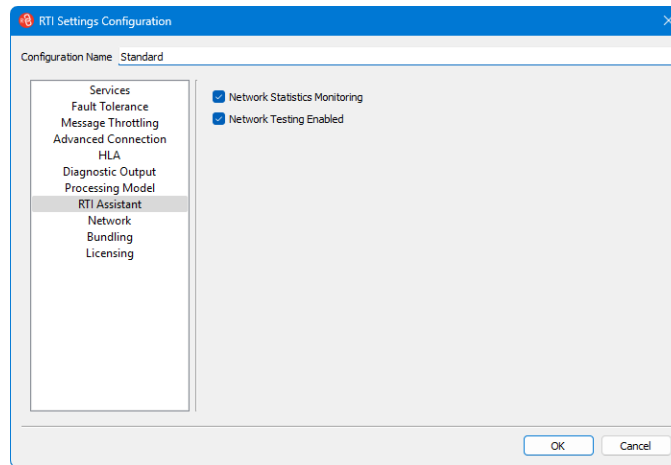
The Network Statistics Monitoring RTI setting specifies whether to publish information for the RTIsPy Federate Statistics and the FOM View. Network statistics monitoring is enabled by default.

► **To edit the Network Statistics Monitoring RTI setting:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔊). A context menu displays.

3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the RTI Assistant page (Figure 28).

Figure 28. RTI Assistant page



8. Select the Network Statistics Monitoring check box.
9. Click OK.

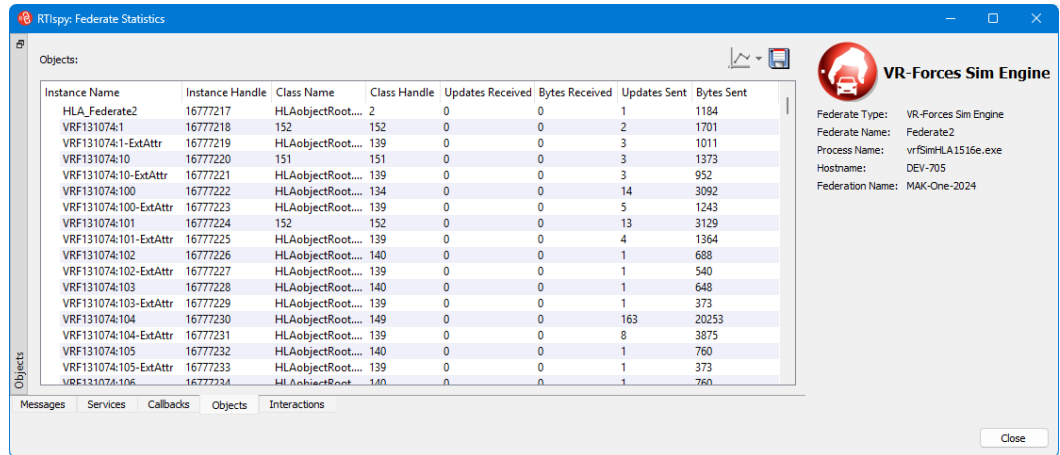
8.3.2 Displaying Object Statistics

The Objects tab lists the objects sent and received by a federate. You can sort the list using any of the columns to make it easier to distinguish the entries.

► **To display object statistics:**

1. In the system tray, right-click the RTI Assistant icon (🔍). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTIsPy: Federate Statistics** or click the RTIsPy: Federate Statistics button (📊). The RTIsPy Federate Statistics window opens.
5. Select the Objects tab (Figure 29).

Figure 29. RTISpy Federate Statistics, Objects tab



Instance Name	Instance Handle	Class Name	Class Handle	Updates Received	Bytes Received	Updates Sent	Bytes Sent
HLA_Federate2	16777217	HLAObjectRoot....	2	0	0	1	1184
VRF131074-1	16777218	152	152	0	0	2	1701
VRF131074-1-ExtAttr	16777219	HLAObjectRoot....	139	0	0	3	1011
VRF131074-10	16777220	151	151	0	0	3	1373
VRF131074-10-ExtAttr	16777221	HLAObjectRoot....	139	0	0	3	952
VRF131074-100	16777222	HLAObjectRoot....	134	0	0	14	3092
VRF131074-100-ExtAttr	16777223	HLAObjectRoot....	139	0	0	5	1243
VRF131074-101	16777224	152	152	0	0	13	3129
VRF131074-101-ExtAttr	16777225	HLAObjectRoot....	139	0	0	4	1364
VRF131074-102	16777226	HLAObjectRoot....	140	0	0	1	688
VRF131074-102-ExtAttr	16777227	HLAObjectRoot....	139	0	0	1	540
VRF131074-103	16777228	HLAObjectRoot....	140	0	0	1	648
VRF131074-103-ExtAttr	16777229	HLAObjectRoot....	139	0	0	1	373
VRF131074-104	16777230	HLAObjectRoot....	149	0	0	163	20253
VRF131074-104-ExtAttr	16777231	HLAObjectRoot....	139	0	0	8	3875
VRF131074-105	16777232	HLAObjectRoot....	140	0	0	1	760
VRF131074-105-ExtAttr	16777233	HLAObjectRoot....	139	0	0	1	373
VRF131074-106	16777234	HLAObjectRoot....	140	0	0	1	760

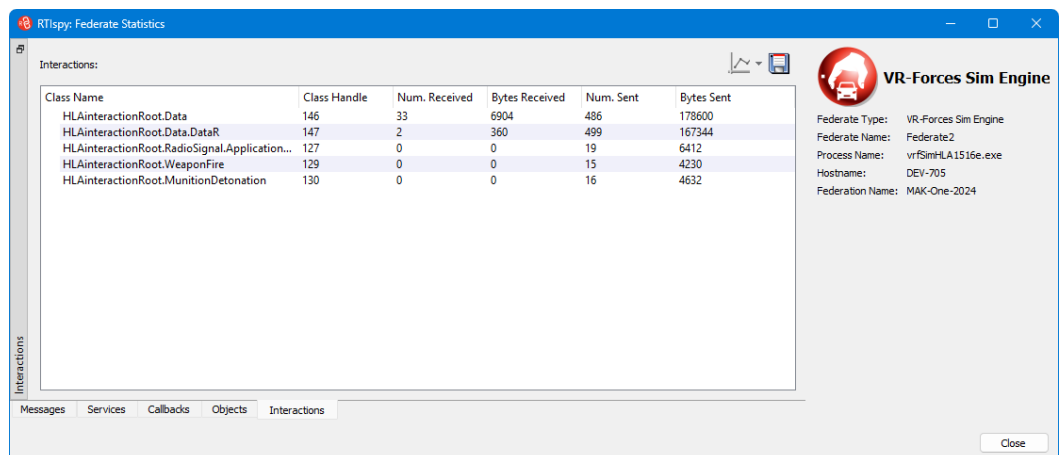
8.3.3 Displaying Interaction Statistics

The Interactions tab lists the interactions sent and received by a federate. You can sort the list using any of the columns to make it easier to distinguish the entries.

► **To display interaction statistics:**

1. In the system tray, right-click the RTI Assistant icon (🔍). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTISpy: Federate Statistics** or click the RTISpy: Federate Statistics button (📊). The RTISpy Federate Statistics window opens.
5. Select the Interactions tab (Figure 30).

Figure 30. RTISpy Federate Statistics, Interactions tab



Class Name	Class Handle	Num. Received	Bytes Received	Num. Sent	Bytes Sent
HLAInteractionRoot.Data	145	33	6904	486	178600
HLAInteractionRoot.Data.DataR	147	2	360	499	167344
HLAInteractionRoot.RadioSignal.Application...	127	0	0	19	6412
HLAInteractionRoot.WeaponFire	129	0	0	15	4230
HLAInteractionRoot.MunitionDetonation	130	0	0	16	4632

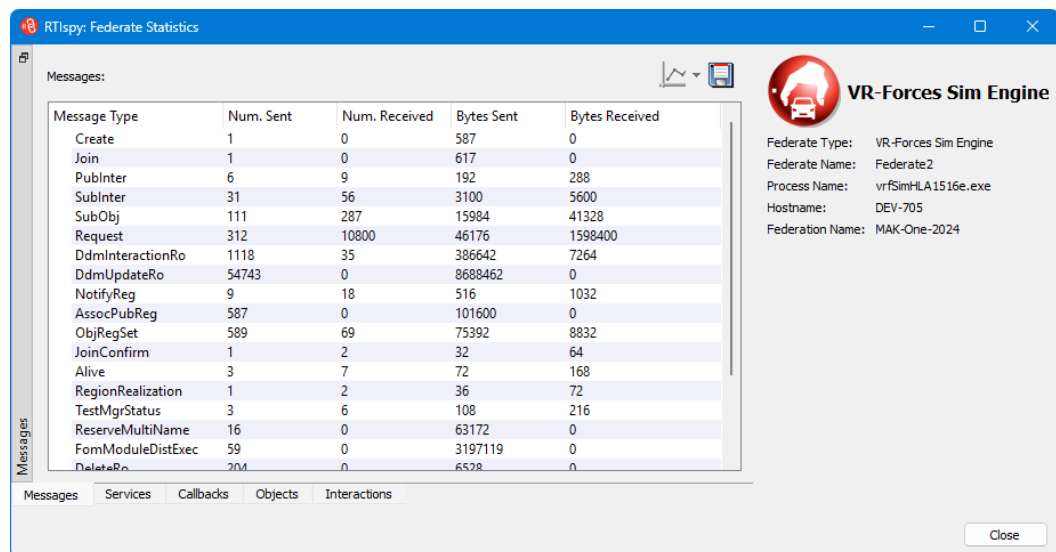
8.3.4 Displaying RTI Message Statistics

The Messages tab lists the RTI message types and shows how often each message type has been sent and received and how large the messages are. It also provides summary statistics on the number of messages sent and received and their sizes.

► **To display a list of RTI message types:**

1. In the system tray, right-click the RTI Assistant icon (🔍). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTISpy: Federate Statistics** or click the RTISpy: Federate Statistics button (📊). The RTISpy Federate Statistics window opens.
5. Select the Messages tab (Figure 31).

Figure 31. RTISpy Federate Statistics, Messages tab



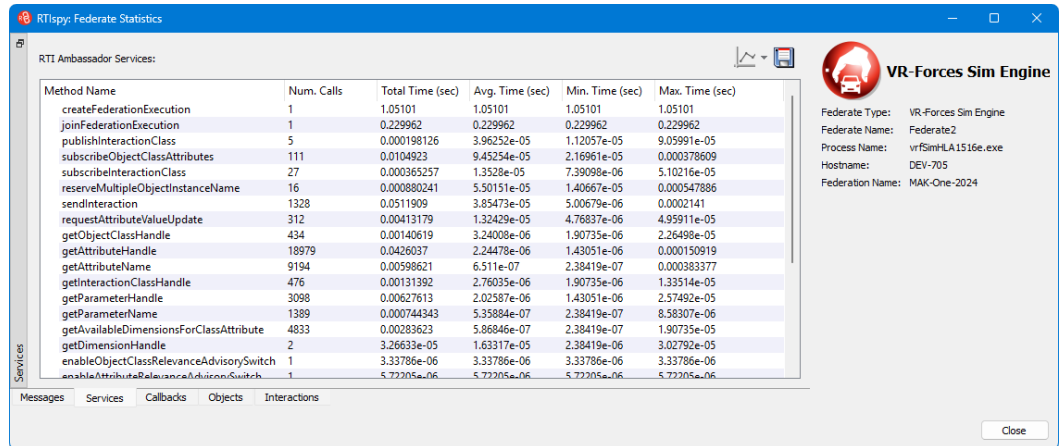
8.3.5 Displaying RTI Services Statistics

The Services tab displays a list of all the calls made to the RTI Ambassador. Observing the amount of time spent on calls that the federate makes can help you locate bottlenecks in your application's use of the RTI and in your callbacks.

► **To display services statistics:**

1. In the system tray, right-click the RTI Assistant icon (🔍). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTISpy: Federate Statistics** or click the RTISpy: Federate Statistics button (📊). The RTISpy Federate Statistics window opens.
5. Select the Services tab (Figure 32).

Figure 32. RTIsPy Federate Statistics, Services tab



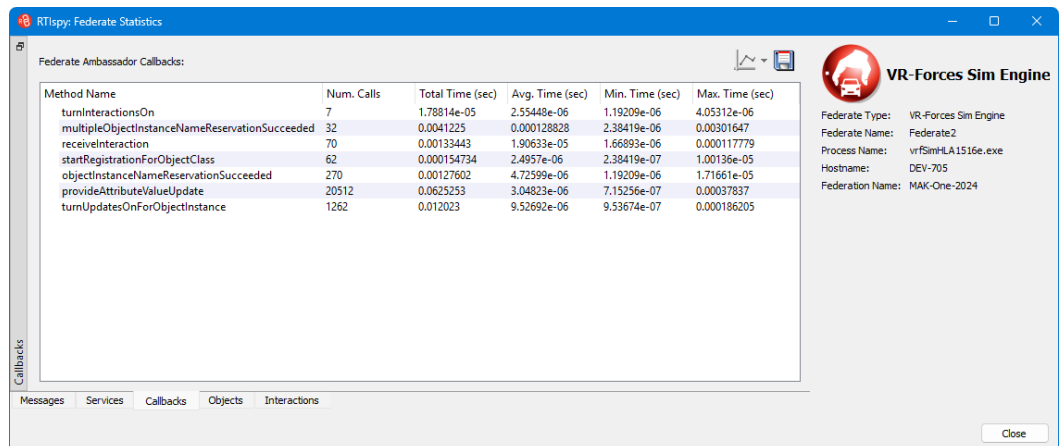
8.3.6 Displaying Callback Statistics

The Callbacks tab lists the callbacks made to the Federate Ambassador.

► **To display callback statistics:**

1. In the system tray, right-click the RTI Assistant icon (🔧). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTIsPy: Federate Statistics** or click the RTIsPy: Federate Statistics button (📊). The RTIsPy Federate Statistics window opens.
5. Select the Callbacks tab (Figure 33).

Figure 33. RTIsPy Federate Statistics, Callbacks tab



8.3.7 Graphing Federate Statistics

You can display graphs of the statistics on the various tabs of the RTIsPy Federate Statistics window. You can graph by item (Figure 34) or rate (Figure 35).

Figure 34. Graphing by item

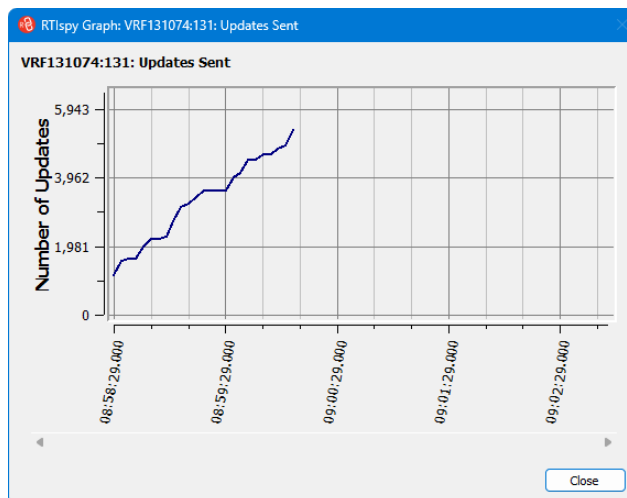
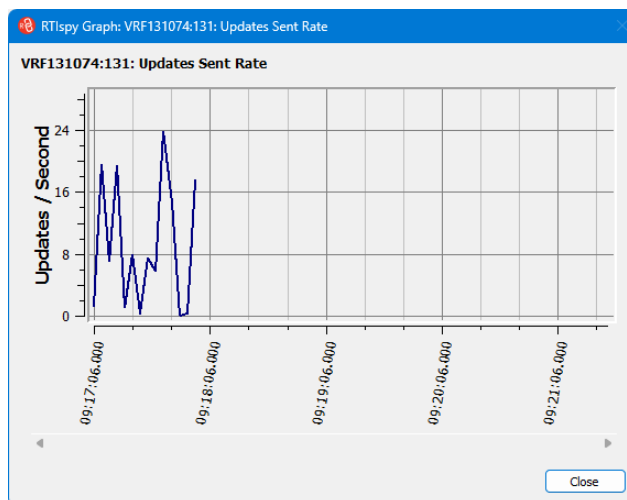


Figure 35. Graphing by rate



► **To graph federate statistics:**

1. In the system tray, right-click the RTI Assistant icon (🔊). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTIsPy: Federate Statistics** or click the RTIsPy: Federate Statistics button (📊). The RTIsPy Federate Statistics window opens (Figure 29 on page 114).
5. Select the tab for which you want to view statistics.
6. Select a row and column.
7. Click the arrow next to the Graph button (📊) on the toolbar. A menu opens.
8. Select an option on the Graph menu. A graph window displays. It plots statistics over time as new data arrives.

8.4. Examples of Using the RTIspy Network Monitoring Tool

The examples in this section show you how you can use the RTIspy Network Monitoring Tool to help resolve common problems.

8.4.1 Finding Bottlenecks Caused by Processing Callbacks

If you suspect that your application is spending too much time processing callbacks, the RTIspy Network Monitoring Tool can help you find the bottlenecks.

► **To locate bottlenecks caused by processing callbacks:**

1. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
2. Choose Federations View. The Federations View opens.
3. Select a federate in the Network Map or Federations list.
4. Choose **Federate > RTIspy: Federate Statistics** or click the RTIspy: Federate Statistics button (). The RTIspy Federate Statistics window opens.
5. Select the Callbacks tab (Figure 33 on page 116).
6. To locate any callbacks that are consuming resources, look at the Federate Ambassador Callbacks list and sort it by Total Time.
7. Review your callback code and look for optimizations. You might consider moving the processing of the callback into a separate thread to isolate the network I/O and simulation processing portions of your code (at the cost of greatly increased complexity).

8.4.2 Identifying the Cause of Slowdowns Under Time Management

Suppose your simulation uses time management to synchronize its federates. If the federates advance through time at different rates, there may be an apparent slowdown in the federates that take larger time steps. For example, if one set of federates requests time advances of 10 seconds per step and a second set requests time steps of 1 millisecond per step, then the 10 second per step federates—which on their own would advance at a rate far greater than real-time—will appear to be slowing down. It may not be obvious what has caused the slowdown. The RTIspy Network Monitoring Tool and the Federations View can help you to identify the problem.

► **To locate slowdowns caused by time management:**

1. In the Federations View, select one of the 10 second per step federates
2. Open the Federate Statistics window and select the Callbacks tab.
3. Sort the RTI Ambassador Callbacks list by Total Time. (Click the heading for the column.) This will show that almost all of the federate's time is spent processing `tick()` calls. This is a sign that your simulation is blocked and waiting for an RTI event to proceed, in this case a Time Advance Grant.
4. In the Federations View, select the federation. You should see the 1 millisecond per-step federates advancing through time while the 10 second per-step federates advance approximately every 10 seconds. This is because they are constrained by

the 1 millisecond federates and can only advance when the 1 millisecond federates have stepped 10,000 times.

8.4.3 Comparing the Costs and Benefits of RTI Features

The RTISpy Network Monitoring Tool can help you to weigh the costs and benefits of various RTI features. Some of the more advanced RTI features, such as MOM or DDM, offer a great deal of functionality. For some simulations, however, the costs in terms of additional message passing and RTI processing outweigh the benefits. The use of DDM, for example, requires the RTI to exchange additional messages for default region information even when simple DM calls are used.

► **To examine the benefits of RTI features:**

1. Run the rtiexec and a simple federate without DDM enabled.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose Federations View. The Federations View opens.
4. Select a federate in the Network Map or Federations list.
5. Choose **Federate > RTISpy: Federate Statistics** or click the RTISpy: Federate Statistics button (). The RTISpy Federate Statistics window opens.
6. Select the Messages tab (Figure 31 on page 115).
7. Run the federate with DDM enabled.
8. Even though DDM has no effect on a simple simulation, you will still see additional messages passed such as AssocPubRegMsgKind, which are required by the DDM implementation.



9. Managing Network Traffic

This chapter describes a variety of ways to configure how the RTI manages network traffic. It does not cover all aspects of network configuration. For a detailed discussion of packet forwarding strategies, including distributed forwarding, see "Chapter 10. Configuring Message Forwarding" on page 135.

9.1. Choosing Best Effort or Reliable Transport	122
9.2. Using the Correct IP Address Format	123
9.3. Packet Bundling for Reliable and Best Effort Messages	123
9.4. Automatically Locating the RTI Forwarder and rtiexec	124
9.5. Sender-Side Filtering (Smart TCP Forwarding)	124
9.5.1 Configuring Sender-Side Filtering	126
9.5.2 Configuring Sender-Side Filtering for rtiexec Messages	127
9.6. Compressing RTI Messages	127
9.7. Enabling Support for Large Attributes and Parameters	128
9.8. Configuring the Network Buffer Sizes	129
9.9. Controlling Congestion for Best Effort Sockets	130
9.10. Configuring the Processing Model	130
9.11. Buffering Automatic Update Requests (1516 Specifications Only)	131
9.12. Filtering by Multicast Groups Using Declaration Management	132
9.12.1 Assigning DM Multicast Groups to Network Interfaces	133

9.1. Choosing Best Effort or Reliable Transport

This section explains how to choose which type of network transport to use.

- 1 – The type of message transport to be used for FOM data and internal messages is set by the FOM Data Transport Type and Send Internal Messages Reliable (TCP) RTI settings. If Send Internal Messages Reliable (TCP) is disabled and FOM Data Transport Type is set to "All Data Best Effort (UDP)", all messages are sent best effort (example 1 in Figure 36 below).

FOM Data Transport Type = Best Effort

Send Internal Messages Reliable (TCP) = 0

- 2 – If you want the RTI's internal bookkeeping messages, such as federate subscription, to be sent using reliable transport, enable Send Internal Messages Reliable (TCP) (example 2 in Figure 36).

FOM Data Transport Type = Best Effort

Send Internal Messages Reliable (TCP) = 1

- 3 – In the FED file, you can specify how you want object attributes and interactions to be sent: reliable transport or best effort. If you want this data sent as stated in your FED file, set FOM Data Transport Type to "Specified by FOM" (example 3 in Figure 36).

FOM Data Transport Type = Use FOM

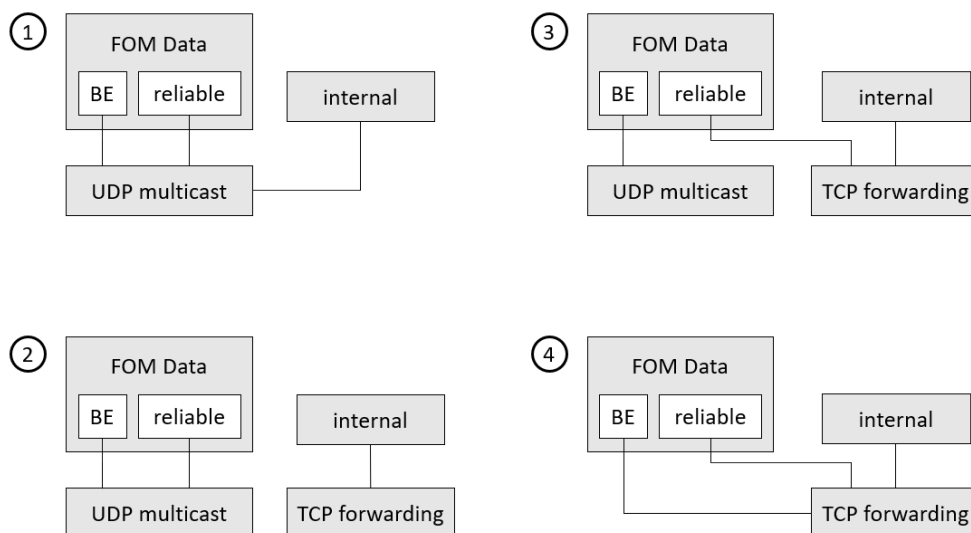
Send Internal Messages Reliable (TCP) = 1

- 4 – If you want the RTI to send all FOM data reliable, regardless of the setting in the FED file, set FOM Data Transport Type to "All Data Reliable (TCP)" (example 4 in Figure 36).

FOM Data Transport Type = Reliable

Send Internal Messages Reliable (TCP) = 1

Figure 36. Configuring the transport type



- BE = best effort
- 0 = disabled
- 1 = enabled

 Figure 36 does not illustrate use of an RTI Forwarder to forward UDP multicast traffic over a WAN. For details, see "10.2. Configuring UDP and Centralized TCP on a WAN" on page 137.

9.2. Using the Correct IP Address Format

The MAK RTI supports IPv4 and IPv6, but you cannot mix the two protocols. If your federation uses IPv4, then all IP addresses in the RTI settings and RTI Assistant must be in the IPv4 format. The same applies to use of IPv6.

9.3. Packet Bundling for Reliable and Best Effort Messages

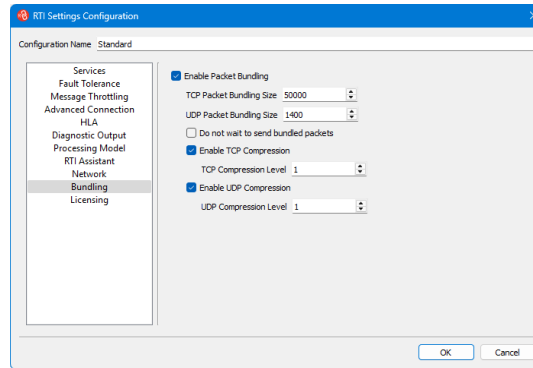
The MAK RTI supports message bundling. If bundling is enabled (with the Enable Packet Bundling RTI setting), the RTI bundles messages into a single TCP packet or UDP datagram, up to the size limit configured by the Packet Bundling Size RTI setting. Message bundling can significantly reduce the amount of bandwidth consumed by packet headers and the processor time spent waiting for network I/O calls. The recommended maximum bundle size is 1492 bytes for WAN traffic, or the maximum transmission unit (MTU) of the local network, if you are just running on a LAN.

 Bundles are flushed at the end of every tick(), even if the bundle is not full.

► To enable packet bundling:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Bundling page (Figure 37).

Figure 37. Bundling page



8. Select the Enable Packet Bundling check box.
9. Enter values in the TCP and UDP Packet Bundling Size boxes.
10. Click OK.

9.4. Automatically Locating the RTI Forwarder and rtiexec

Using multicast discovery, the RTI can automatically locate an RTI Forwarder or rtiexec running on the LAN (that is, without specifying **RTI_tcpForwarderAddr**).

- To enable multicast discovery, set **RTI_mcastDiscoveryEnabled** to 1.

With multicast discovery enabled, federates send a beacon to the multicast group selected by the **RTI_mcastDiscoveryAddrString** parameter. If an RTI Forwarder is running on the LAN, it responds with the correct address. The federate will continue trying to discover a forwarder until it reaches the limit specified by the **RTI_mcastDiscoveryTries** parameter. At that point, it defaults to best effort only.

You can use multicast discovery to ensure that only one RTI Forwarder and only one rtiexec are running on a LAN. This is particularly useful in cases where the rtiexec may run on different hosts from execution to execution.

9.5. Sender-Side Filtering (Smart TCP Forwarding)

Sender-side filtering (also called smart TCP forwarding) minimizes bandwidth by taking advantage of subscription information down to the level of each attribute instance, including taking DDM into account. If an attribute is not in scope for a receiving federate, it can be omitted from the transmission. Subscriptions are tracked at the federate level and a routing mask is appended to each update and interaction message. The routing mask is created based on the maximum number of federates permitted, as specified by the Max Number of Federates RTI setting.

As you increase the number of federates permitted, the update and interaction message size increases. This leads to a possible trade-off between the cost of the small increase in message size and the lowered bandwidth achieved because entire transmissions may be filtered out.

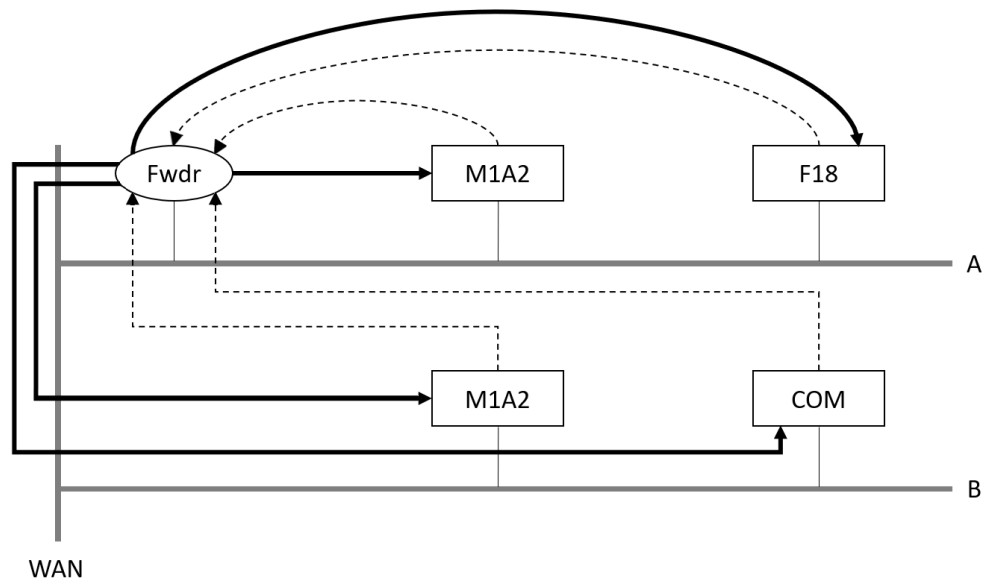
Sender-side filtering is beneficial only if federates subscribe to a subset of the attributes being published. If a federate subscribes to all data, sender-side filtering can decrease performance. Figure 38 and Figure 39 on the next page compare TCP forwarding to sender-side filtering. In those figures:

- The dashed arrows show the federate to forwarder.
- The solid arrows show the forwarder to the federate. The line thickness illustrates the relative amount of traffic (1:3).

TCP Forwarding Without Filtering

When you use TCP forwarding without filtering, all federates receive all messages, all of which must travel over the LAN and WAN to the RTI Forwarder then back to the other federates. Each federate sends updates to the RTI forwarder, which forwards them to all other federates, including other federates on the local LAN.

Figure 38. TCP forwarding



Sender-Side Filtering

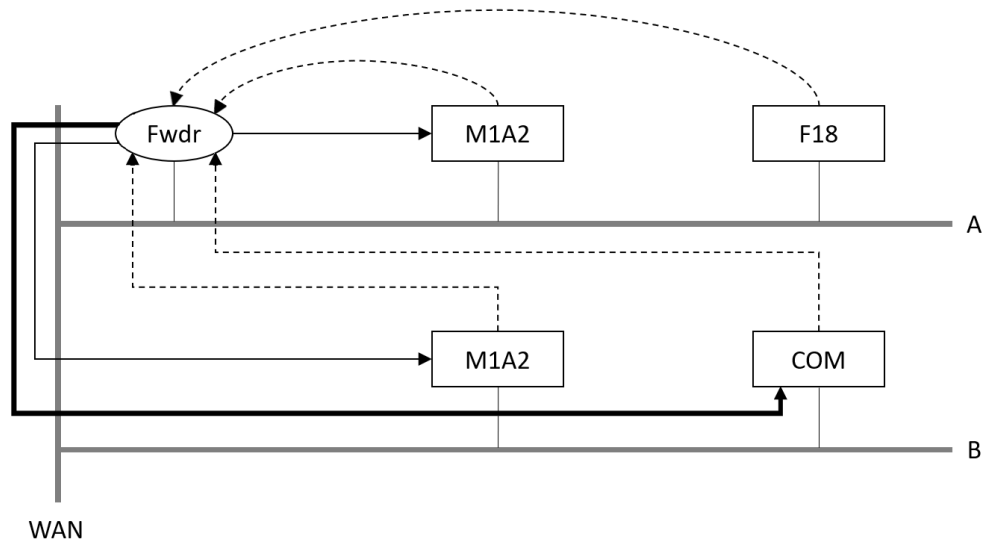
When you use sender-side filtering, all federates still send updates through the RTI Forwarder. However, they receive updates based on their subscriptions; only updates of interest are sent to receiving federates.

In Figure 39 on the next page, M1A2 federates are subscribed only to ground entity updates, and therefore they only receive updates from other M1A2 federates, in this case one update each. The F18 federate is subscribed to and only receives updates from other fixed-wing entities, in this case, none. The COM federate is subscribed to and receives all updates.



- If you use sender-side filtering, the rtiexec can support only one federation at a time.
- Sender-side filtering is not compatible with fixed grid DDM.

Figure 39. Sender-size filtering



9.5.1 Configuring Sender-Side Filtering

To configure sender-side filtering, edit the Smart Forwarding Level RTI setting and the Max Number of Federates setting.

Smart Forwarding Level specifies the TCP forwarding mode:

- No Smart Forwarding. Basic TCP forwarding (no subscription-based routing or filtering).
- Subscription Routing. Enable subscription based routing.
- Routing + Server-Side Filtering. Enable subscription based routing and disable send for updates and interactions that have no subscribers.

Default: No Smart Forwarding.

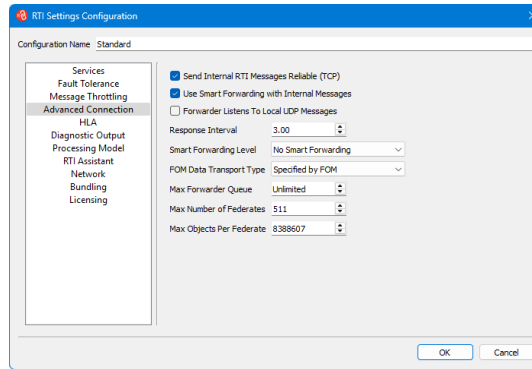
Max Number of Federates specifies the maximum number of federates allowed in a federation at any one time. This parameter only applies when sender-side filtering is enabled. Each update and interaction message has a bitmask of at least this number of bits appended, so increasing the number of federates increases the overhead. Default: 511.

► **To configure sender-side filtering:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.

6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Advanced Connection page (Figure 40).

Figure 40. Advanced Connection page



8. Select an option from the Smart Forwarding Level list.
9. Optionally, change the value for Max Number of Federates.
10. Click OK.

9.5.2 Configuring Sender-Side Filtering for rtiexec Messages

By default, the RTI Forwarder sends internal messages only to the rtiexec (but not to federates), and FOM data only to federates (but not to the rtiexec). This filtering can significantly reduce the number of internal message transmissions, reducing both bandwidth and network overhead. In those federations where internal RTI messages consume a significant portion of network resources, this can lead to a substantial, federation-wide performance improvement. You enable or disable this message filtering with the `RTI_enableFedexMsgRouting` parameter.

9.6. Compressing RTI Messages

The RTI supports compression and decompression of RTI messages. You can enable compression separately for UDP (best effort) and TCP (reliable). The amount of compression applied can also be controlled with separate configuration parameters. Compression can be configured separately for each federate, rtiexec, and forwarder in your federation.

To configure compression, use these RTI settings:

- **Enable TCP Compression.** Enables compression of RTI message payloads transmitted over TCP connections. Consistency checking is enforced on this parameter.

- **TCP Compression Level.** Specifies the amount of compression applied to RTI message payloads over TCP connections:
 - 1 – minimal compression.
 - 2 through 8 – increasing levels of compression.
 - 9 – maximum compression.
- **Enable UDP Compression.** Enables compression of RTI message payloads transmitted in UDP datagrams. Consistency checking is enforced on this parameter.
- **UDP Compression Level.** Specifies the amount of compression applied to RTI message payloads in UDP datagrams, using the same options as TCP Compression Level.

 If you want to use the RTIspy API to override the socket compression implementation, please contact MAK technical support for assistance.

► **To configure compression in RTI settings:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Bundling page (Figure 37 on page 124).
8. Select the Enable TCP Compression or Enable UDP Compression check box, or both.
9. Enter a values in the appropriate compression level box.
10. Click OK.

9.7. Enabling Support for Large Attributes and Parameters

The **RTI_use32BitsForValueSize** parameter enables or disables support for sending update and interaction messages that have attributes and parameters that are larger than 64 KB. The default message format uses 16-bit integers (maximum message size 64 KB) to represent the size of attribute and parameter data elements. When you enable this parameter, the message format is modified to use 32-bit integers for each value. Any resulting large messages (> 64KB) must be sent using TCP. The fragmentation and reassembly of messages is only supported by the RTI reliable connection that uses TCP. The best effort connection does not support data fragmentation other than what is provided by the underlying UDP/IP, and it is limited to 64K packets. You can send messages using best effort when this parameter is enabled, but the messages must be smaller than 64K.

You must adjust the network buffer sizes to accommodate the largest message that will be transmitted.

9.8. Configuring the Network Buffer Sizes

You can configure the TCP and UDP network receive buffers, the send buffer, and the TCP buffer size. If you enable big messages, you must adjust the network buffer sizes to accommodate the largest message that will be transmitted.

The Socket Receive Buffer Size RTI setting configures the size of the TCP and UDP network receive buffers. Increasing the size can increase the RTI's tolerance of peak network loading. This buffer size should be large enough to hold several of the largest messages. Default: 2000000.

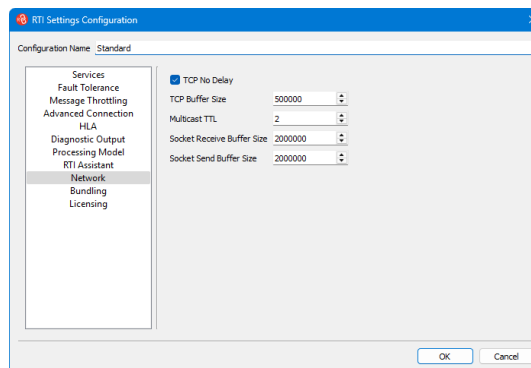
The Socket Send Buffer Size RTI setting configures the socket send buffer size. This buffer size should be large enough to hold the largest message to be sent. Default: 200000.

The TCP Buffer Size RTI setting sets the size of the internal buffer into which TCP packets get read. This buffer size should be large enough to hold the largest message to be received. Default: 500000.

► To edit the network buffer size RTI settings:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Network page (Figure 41).

Figure 41. Network page



8. Type values in the Socket Receive Buffer Size box, the Socket Send Buffer Size box, and the TCP Buffer Size box.
9. Click OK.

9.9. Controlling Congestion for Best Effort Sockets

When **RTI_enableBestEffortSendRetry** is enabled, the RTI tries to provide limited congestion control for best effort sockets. To configure congestion control, set **RTI_bestEffortSendRetries** and **RTI_bestEffortSendRetryWaitUsec**. **RTI_bestEffortSendRetries** specifies the number of send attempts before discarding the message. **RTI_bestEffortSendRetryWaitUsec** specifies the number of microseconds to wait after each send attempt for a blocked socket.

RTI_bestEffortSendRetries and **RTI_bestEffortSendRetryWaitUsec** are advanced network configuration parameters. When used together, they provide limited congestion control for best effort sockets. **RTI_bestEffortSendRetries** specifies the number of send attempts before discarding the message. **RTI_bestEffortSendRetryWaitUsec** specifies the number of microseconds to wait after each send attempt for a blocked socket.

9.10. Configuring the Processing Model

The Processing Model RTI setting specifies the processing model, as follows:

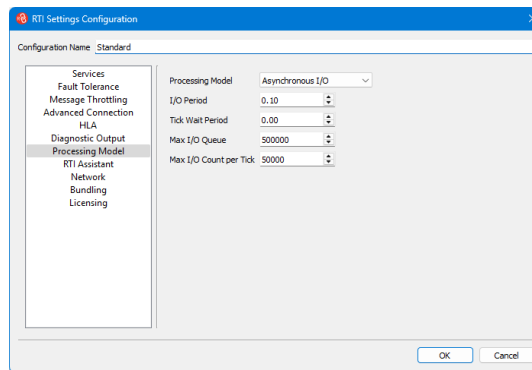
- Synchronous – synchronous mode.
- Asynchronous I/O (default). The RTI processes network reads and writes in a secondary thread. Asynchronous I/O can improve performance in federations that often produce bursts of network traffic and may reduce the number of dropped messages.
- Asynchronous Processing. The RTI processes messages in the asynchronous thread, but it does not invoke any resulting callbacks. This processing mode may improve the response and efficiency of some distributed algorithms (for example, time management). However, it requires the caching of callback parameters, which results in additional memory allocation and data copy operations.
- Asynchronous Callbacks. The RTI performs federate callbacks in the asynchronous thread. Asynchronous callbacks may offer performance benefits to certain classes of federate. However, asynchronous callbacks significantly increase the complexity of federates, because the federate designer must ensure that all calls to the Federate Ambassador are fully thread-safe.

► To edit the Processing Model RTI setting:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.

4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Processing Model page (Figure 42).

Figure 42. Processing Model page



8. Select an option from the Processing Model list.
9. Click OK.

9.11. Buffering Automatic Update Requests (1516 Specifications Only)

Federates using any version of the 1516 specification (HLA 1516, HLA Evolved, and HLA 4) can instruct the RTI to buffer automatic update requests temporarily, to accumulate automatic requests from multiple federates. This can reduce the number of automatic `provideAttributeUpdate` callbacks required for changes in publication and subscription. To enable this feature, enable the `Enable Message Throttling` RTI setting and set `Provide Delay` to the number of seconds the RTI should delay before responding to requests.

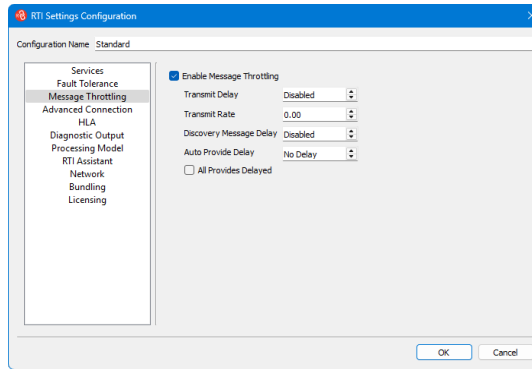
To buffer all update requests, not just automatic requests, you can enable the `All Provides Delayed` RTI setting.

► To edit the RTI settings for buffering automatic update requests:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.

6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Message Throttling page (Figure 43).

Figure 43. Message Throttling page



8. Select the Enable Message Throttling check box.
9. Enter a value in the Auto Provide Delay box or set it to No Delay.
10. Optionally, if you specified a value for Auto Provide Delay, select the All Provides Delayed check box.
11. Click OK.

9.12. Filtering by Multicast Groups Using Declaration Management

You can assign object and interaction classes to separate multicast groups to enable basic filtering by multicast groups using declaration management (DM) services. Use the RID file parameters, **RTI-addDMObjectMulticastAddr** and **RTI-addDMInteractionMulticastAddr**.

The syntax for the parameters is:

```
(RTI-addDMObjectMulticastAddr "ObjectClassName" "MulticastAddress"
  "NestingOperator" "interface")
(RTI-addDMInteractionMulticastAddr "InteractionClassName"
  "MulticastAddress" "NestingOperator" "interface")
```

where:

- *ObjectClassName* is a fully qualified FOM class of the form "*ObjectRoot.BaseEntity.Platform*".
- *MulticastAddress* is a valid multicast address of the form "229.7.7.9" for IPv4 or "ff12::7:10" for IPv6.
- *NestingOperator* is a quoted string containing either **exclusive** or **inclusive**. The nesting operator values determine whether the assignment of the multicast address will apply to the specific class exclusively or to that class and all derived classes.

- *interface* specifies a network interface. For details, see "9.12.1 Assigning DM Multicast Groups to Network Interfaces" below.

Organize entries in the order of base classes to leaf classes. Exclusive entries must be placed after all inclusive entries from which the class inherits.

In the following example, the address 229.7.7.9 is assigned to all classes derived from "ObjectRoot.BaseEntity.Platform" except for "ObjectRoot.BaseEntity.Platform.GroundVehicle", which is assigned 229.7.7.10:

```
(RTI-addDMObjectMulticastAddr "ObjectRoot.BaseEntity.Platform"
  "229.7.7.9" "inclusive" "0.0.0.0")
(RTI-addDMObjectMulticastAddr "ObjectRoot.BaseEntity.Platform.
  GroundVehicle" "229.7.7.10" "exclusive" "0.0.0.0")
```

When using these parameters:

- A federate subscription causes the LRC to join all multicast groups specified for the subscribed class and its derived classes. For example, given the previous example, a subscription to ObjectRoot.BaseEntity.Platform would result in the LRC joining multicast groups 229.7.7.9 and 229.7.7.10 on 0.0.0.0, the default interface.
- The LRC transmits object attributes to the multicast group specified by the class of the object instance (not the class where the attributes are defined). Given the previous example, the LRC transmits a WorldLocation attribute (defined in class BaseEntity) of an object instance of class ObjectRoot.BaseEntity.Platform.GroundVehicle to the multicast group 229.7.7.10. It would transmit the same attribute of an object instance of class ObjectRoot.BaseEntity.Platform to the multicast group 229.7.7.9.
- The LRC transmits interactions to the multicast group specified by the interaction class.

 The functionality described in this section is separate from any filtering that the DDM services might perform.

9.12.1 Assigning DM Multicast Groups to Network Interfaces

The **RTI-addDMObjectMulticastAddr** parameter has a parameter that assigns DM multicast groups to specific network interfaces. If you have multi-homed machines, this parameter allows you to segment RTI traffic onto multiple physical networks without modifying your system's routing tables. For example, if a multi-homed machine has network interfaces with address 10.0.1.211 and 192.168.1.211, you can send all HLA object traffic to the 10.0.1.x network:

```
(RTI-addDMObjectMulticastAddr "ObjectRoot" "224.0.0.1" "inclusive"
  "10.0.1.211")
```

The **RTI-addDMInteractionMulticastAddr** parameter has a similar parameter. For example, if a multi-homed machine has network interfaces with address 10.0.1.211 and 192.168.2.211 you can send all HLA interaction traffic to the 192.168.1.x network using the setting:

```
(RTI-addDMInteractionMulticastAddr "InteractionRoot" "224.0.0.2"
  "inclusive" "192.168.2.211")
```

 The multicast address and network interface must be of the same address family, either IPv4 or IPv6.



10. Configuring Message Forwarding

The MAK RTI offers a variety of options for configuring how messages are forwarded to federates. The first part of this chapter discusses networking on LANs and WANs using UDP or centralized forwarding. The rest of the chapter describes how to configure distributed forwarding. Use of distributed forwarding can significantly reduce network traffic over a LAN or WAN.

10.1. Configuring UDP and Centralized TCP Forwarding on a LAN	136
10.1.1 Configuring Best Effort (UDP) on a LAN	136
10.1.2 Configuring Centralized TCP Forwarding on a LAN	136
10.2. Configuring UDP and Centralized TCP on a WAN	137
10.2.1 Configuring Centralized TCP Forwarding	137
10.2.2 Configuring UDP Over a WAN Without Distributed Forwarding	138
10.3. Distributed Forwarding Topologies on a LAN	139
10.3.1 Forwarder Groups	139
10.3.2 Singleton Forwarders	140
10.3.3 Advanced Distributed Forwarding on a LAN	141
10.4. Distributed Forwarding Topologies on an WAN	142
10.4.1 Optimizing Distributed Forwarding across a WAN	143
10.5. Configuring Federates for Distributed Forwarding	146
10.6. Configuring RTI Forwarders for Distributed Forwarding	146
10.6.1 Configuring RTI Forwarders Using the RTI Assistant	146
10.6.2 Creating a Routes File	147
10.6.3 UDP Forwarding	149
10.6.4 Configuring Single-Portal Forwarder Groups	150
10.6.5 Configuring Load-Balanced Forwarder Groups	151
10.6.6 Configuring Distributed UDP Forwarding	152
10.7. Connecting to Remote Forwarders using the RTI Assistant	154
10.7.1 Connecting to Remote Forwarders on the Command Line	155
10.8. Configuring Routers or Firewalls for WAN Federations	155
10.9. Special Distributed Forwarder Networks	155

10.9.1 Configuring Hierarchical Forwarder Networks 155

10.1. Configuring UDP and Centralized TCP Forwarding on a LAN

As discussed in "2.3. Network Topologies for RTI Messages" on page 29, the MAK RTI supports sending messages over a LAN via best effort, centralized forwarding, and distributed forwarding. For details about configuring distributed forwarding, see "10.3. Distributed Forwarding Topologies on a LAN" on page 139 and the sections that follow it.

 The MAK RTI supports IPv4 and IPv6, but you cannot mix the two protocols. If your federation uses IPv4, then all IP addresses in the RTI settings and RTI Assistant must be in the IPv4 format. The same applies to use of IPv6.

10.1.1 Configuring Best Effort (UDP) on a LAN

To configure best effort communications, set **RTI_fomDataTransportTypeControl** to 0 or 1, or set **RTI_internalMsgReliableWhenUsingRtiexec** to 0, as described in "9.1. Choosing Best Effort or Reliable Transport" on page 122. Optionally, specify:

- **RTI_udpPort**. Specifies the port number to be used for best effort messages sent among federates. Default: 4000.
- **RTI_destAddrString**. Specifies the IP address to which best effort federation execution traffic is sent. This will typically be a broadcast address or multicast address. Default: "229.7.7.7" (a multicast address). A value of "0.0.0.0" is a special value that indicates that the default broadcast address should be used. If any of your federates are running on platforms on which multicast is not supported, you must use a broadcast address rather than a multicast address.

10.1.2 Configuring Centralized TCP Forwarding on a LAN

By default, the RTI uses reliable transport and starts an RTI Forwarder when you start the rtiexec. In other words, the configuration instructions in this section are the default behavior of the RTI when you run an rtiexec using a connection configuration in the RTI Assistant.

► **To configure centralized TCP forwarding:**

1. Enable use of reliable transport for internal messages or FOM data or both, by setting **RTI_fomDataTransportTypeControl** to 0, or **RTI_internalMsgReliableWhenUsingRtiexec** to 1. Optionally, you can set **RTI_fomDataTransportTypeControl** to 2 to force all FOM data to be sent reliably, regardless of the settings in the FED file.
2. Set **RTI_tcpForwarderAddr** to the IP address of the machine that is running the RTI Forwarder.
3. Set **RTI_tcpForwarderAddr** to the IP address of the machine that is running the RTI Forwarder.
4. Set **RTI_tcpPort** to the port to be used for TCP connections by adding the following line to the RID file:

```
setqb RTI_tcpPort port
```

You must run the RTI Forwarder if you are using reliable transport (set **RTI_useRtiExec** to 1).

For example, if you want to use reliable transport for your internal messages and FOM data, and the machine that you are running the rtiexec on has the IP address 192.168.x.x, then your RTI initialization data (RTI settings) file should include the following:

```
(setqb RTI_useRtiExec 1)
(setqb RTI_internalMsgReliableWhenUsingRtiexec 1)
(setqb RTI_fomDataTransportTypeControl 0)
(setqb RTI_tcpForwarderAddr "192.168.x.x")
```

i You can set the size of the buffer into which TCP packets get read. To set the buffer size, use the TCP Buffer Size (**RTI_tcpBufferSize**). The default size is 500000.

10.2. Configuring UDP and Centralized TCP on a WAN

You can use best effort or reliable transport over a WAN. The MAK RTI supports various packet forwarding strategies for reducing the amount of bandwidth consumed in a WAN environment. The RTI Forwarder supports centralized TCP forwarding, distributed TCP forwarding, and distributed UDP forwarding. For details about configuring distributed forwarding on a WAN, see "10.3. Distributed Forwarding Topologies on a LAN" on page 139 and the sections that follow it.

10.2.1 Configuring Centralized TCP Forwarding

You can use the MAK RTI's centralized TCP forwarding scheme in a WAN environment. This configuration is relatively simple to set up: as you would in a LAN, just set **RTI_tcpForwarderAddr** to the address of the RTI Forwarder.

Although centralized TCP forwarding is easy to configure, it is not particularly efficient. When you only have one RTI Forwarder, all TCP messages in the federation must travel through that central RTI Forwarder, even when both the sender and ultimate recipient are located on one LAN, and the RTI Forwarder is located on another. Obviously, this results in many more messages being sent across WAN links than are necessary. Therefore, the preferred method for running federations over a WAN is distributed forwarding.

Configuring Routers or Firewalls to Support WAN-based Federations

When you run a federation across a WAN, it is possible, and even likely, that the remote networks will be protected by firewalls or routers that perform Network Address Translation (NAT). The RTI Forwarder must be accessible to all other federate hosts or forwarders via the designated distributed forwarder port. Therefore, you will probably need to do some additional configuration to allow traffic to get through the firewalls to your RTI Forwarders. There are several options for doing this, including NAT with port forwarding, using a demilitarized zone (DMZ), virtual private networks (VPNs), static routing, and virtual LANs. You should consult with your network administrator and the administrators of other networks in your exercise to determine the best method to use. This section explains the port forwarding approach.

► If you are using centralized TCP forwarding, you only need to configure one firewall: the one on the RTI Forwarder's LAN:

1. Make sure that all remote federates are configured to send TCP packets to the firewall's IP address, rather than the IP address of the RTI Forwarder's machine itself (which will not be reachable directly). In other words, set **RTI_tcpForwarderAddr** to the IP address of the firewall protecting the RTI Forwarder's LAN.
2. Configure that firewall so that it will forward TCP packets received on the RTI's TCP port to the machine that hosts the RTI Forwarder. (If you are using centralized TCP forwarding, then the RTI's TCP port will be the same as the RTI's UDP port).

If you are using distributed forwarding, see "10.8. Configuring Routers or Firewalls for WAN Federations" on page 155 for configuration details.

10.2.2 Configuring UDP Over a WAN Without Distributed Forwarding

The preferred method of supporting best effort traffic over a WAN is to have a local RTI Forwarder listen to local multicast traffic, then forward those packets to remote RTI Forwarders via TCP. If you do not want to use the distributed forwarding method for UDP messages, you can use the strategy described in this section.

In this configuration, multiple copies of each message get sent over the WAN. The MAK RTI supports the specification of multiple destination addresses for best effort packets. This is often necessary when communicating across a WAN, since most routers do not forward multicast packets.

 If you are running two or more federates on the same computer and that computer is not available by multicast, then you must use the RTI Forwarder for best effort. This is because unicast UDP packets are only available to one application per IP:port pair. In other words, the procedure described in this section does not work if you run multiple federates on a computer.

To use this method, set the **RTI_distributedUdpForwarderMode** parameter to 0. Use **RTI_addDestAddrString** to add one or more destination addresses. All best effort data will be sent to each registered address. Configure a separate RID file for each LAN, such that a unicast address for each remote federate (or computer if a computer hosts multiple federates) is added as a destination. The default destination address (**RTI_destAddrString**) is automatically added to the list of destination addresses for LAN best effort communication, so do not add local federates.

For example, suppose two LANs have federates at the following IP addresses:

- LAN 1:
 - 207.86.232.1
 - 207.86.232.2
 - 207.86.232.3

- LAN 2:
208.86.232.1
208.86.232.2
208.86.232.3

Add the following entries to the RID files:

- LAN 1 RID file
(RTI-addDestAddrString "208.86.232.1")
(RTI-addDestAddrString "208.86.232.2")
(RTI-addDestAddrString "208.86.232.3")
- LAN 2 RID file
(RTI-addDestAddrString "207.86.232.1")
(RTI-addDestAddrString "207.86.232.2")
(RTI-addDestAddrString "207.86.232.3")

10.3. Distributed Forwarding Topologies on a LAN

For very large federations that send large amounts of reliable data, a single RTI Forwarder may eventually limit performance of the federation because it is responsible for sending a copy of each message to each federate. Running multiple RTI Forwarders can relieve this problem by allowing the burden of message distribution to be balanced among the forwarders. This preserves the benefits of employing RTI Forwarders while reducing the possibility of a forwarder overload.

If you use multiple RTI Forwarders, you can configure them in forwarder groups or as individual (singleton), fully connected forwarders. You can combine these two methods, where appropriate. For a discussion of when each alternative may be advantageous, see "10.3.3 Advanced Distributed Forwarding on a LAN" on page 141.

i If you run multiple forwarders on a LAN-only configuration, you must disable distributed UDP forwarding. (Set **RTI_distributedUdpForwarderMode** to 0.) There is no benefit to distributed UDP forwarding in this situation because multicast and broadcast over a LAN are more efficient.

10.3.1 Forwarder Groups

You can set up RTI Forwarders to share LAN traffic by configuring them into a forwarder group. A forwarder group consists of a set of RTI Forwarders and all of the federates associated with them.

In a forwarder group, each federate is assigned to a primary forwarder. Federates send messages only to their primary forwarder. The primary forwarder relays the messages to all of the other federates in the forwarder group. Federates receive messages from all of the forwarders in the group. To achieve a good load sharing arrangement, you must assign federates to primary forwarders in a way that assures proper load dispersal. The forwarders in a group can also communicate with each other, primarily to exchange internal messages and messages destined for the rtiexec (which only maintains a connection to a single forwarder in the group).

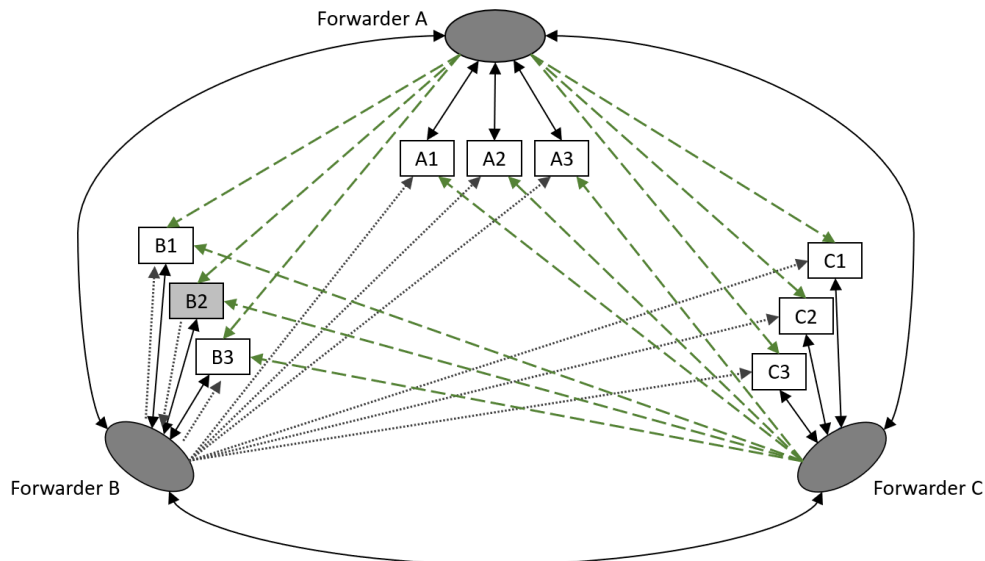
Figure 44 shows a forwarder group consisting of three RTI Forwarders, each of which serves several federates. The solid bi-directional lines represent connections between federates and their primary forwarders. The dashed uni-directional lines represent connections from the other (ancillary) forwarders. Federate B2 (shaded) sends a message to its primary RTI Forwarder (Forwarder B), which in turn sends it to all the other federates in the group (shaded lines).

i The primary and ancillary forwarders are different for each federate. Forwarder A is the primary forwarder for federate A1, but is an ancillary forwarder to federate B1.

In the diagram:

- Double-headed black arrows represent federate to primary forwarder.
- Dashed green arrows represent the ancillary forwarder.
- Dotted gray arrows represent the path of example message.

Figure 44. Distributed forwarding on a LAN with a forwarder group



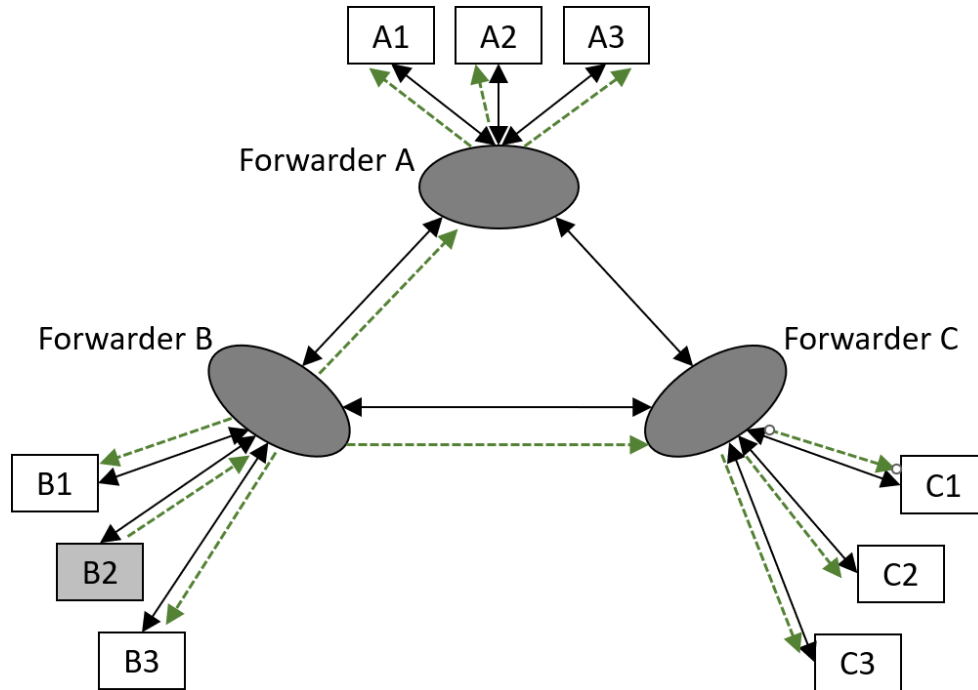
10.3.2 Singleton Forwarders

If you do not want to create forwarder groups, you can connect federates to individual forwarders. These singleton forwarders can then be connected in a fully-connected configuration as shown in (Figure 45). This is known as a distributed singleton forwarder network. Each forwarder and its associated federates are essentially treated as a forwarder group that has one forwarder. The difference between this configuration and the forwarder group configuration is that the federates do not connect to any ancillary forwarders. So a messages sent by a federate (shaded federate and lines in Figure 45) must be relayed by its forwarder to the other forwarders, which then send it to the federates they are connected to.

In the diagram:

- Double-headed black arrows represent bidirectional connections.
- Dashed green arrows represent the path of example message.

Figure 45. Distributed Singleton Forwarder Network on a LAN



10.3.3 Advanced Distributed Forwarding on a LAN

Forwarder groups and singleton forwarder networks both spread LAN traffic among multiple RTI Forwarders, thereby reducing the potential bottleneck of a centralized forwarder setup. In choosing which topology to use for your LAN, consider the following:

- Since group forwarders relay messages directly to the other federates on the LAN, group forwarding reduces latency (in most cases) by eliminating the hop between forwarders that is necessary if the distributed singleton forwarder network is used.
- Depending on the number of federates in a federation, a singleton forwarder network can have advantages over a forwarder network comprised of a single forwarder group.

Consider the case of a federation with a very large number of federates, for example 3000. If all of these are in a single forwarder group of 30 forwarders, in which each RTI Forwarder acts as the primary forwarder for 100 federates, each message from a federate must be retransmitted 2999 times by a primary forwarder in order to reach all 3000 federates. Since the primary forwarder likely only has one network interface over which to send these messages, the average latency is calculated as:

$$t_{msg} \times (\# \text{ of federates} / 2)$$

where t_{msg} is the time required to transmit the message once over the LAN.

In contrast, consider a distributed singleton forwarder network and the time required to distribute a message. The primary forwarder only has to relay the message 128 times (99 copies to the other federates it is connected to and 29 copies to the other RTI Forwarders). Since individual forwarders can probably transmit simultaneously if they are on a switched Ethernet backbone, all 30 forwarders could theoretically be distributing the message simultaneously to the federates connected to them. Potentially, the average latency is reduced to:

$$(t_{msg} \times (\# \text{ of forwarders} / 2)) + (t_{msg} \times (\# \text{ of federates per forwarder} / 2))$$

The number of federates per forwarder is an order of magnitude less than the total number of federates. The time it takes for each forwarder to distribute the message to all federates connected to it is the largest contributor to the overall latency. Therefore, the latency should be reduced by an order of magnitude.

There is another way to configure large single-LAN federations. You can use one of the distributed forwarding methods used for sending reliable traffic over a WAN. Since the only difference between reliable communication over a LAN and over a WAN is the availability of bandwidth, it may be advantageous to use multiple forwarder groups on a LAN. Each group would consist of multiple RTI Forwarders that share the load and then interconnect over the LAN in a load-balanced configuration. This allows RTI Forwarders in disparate groups to handle message relay simultaneously within each group, thus potentially reducing latency even further.

There is no one correct topology. You must consider the unique circumstances of your federation when determining the best strategy for optimizing communication.

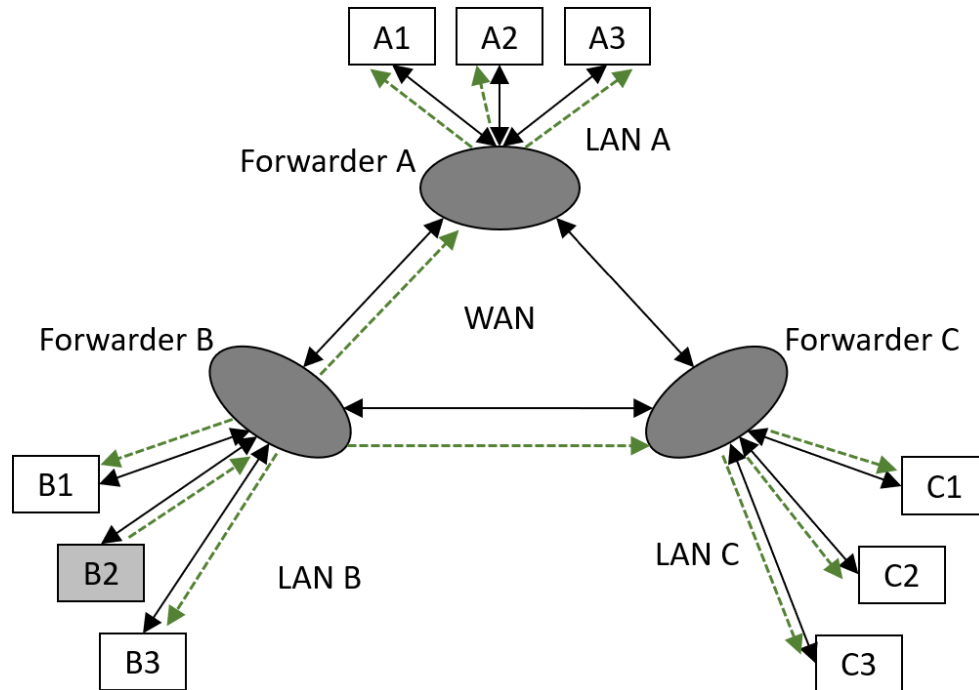
10.4. Distributed Forwarding Topologies on an WAN

The simplest distributed forwarding configuration for a WAN is to use individual (singleton), fully connected forwarders at each LAN. The forwarder at each LAN forwards messages to its local federates and between forwarders on remote LANs. The forwarders handle both reliable (TCP) and best effort (UDP) messages.

Similar to the centralized forwarder strategy, the sending federate passes the message only to the RTI Forwarder to which it is connected and the forwarder re-sends the message to all other federates connected directly to it (its federation subset). However, the RTI Forwarder also sends a copy of the message to all remote RTI Forwarders whose federation subsets contain federates that require the message. The remote forwarders distribute the message to all interested federates in their subset. Figure 46 shows a distributed forwarding network. This figure is essentially identical to Figure 45 on the previous page. The only difference is that now the RTI Forwarders are on separate LANs.

In the diagram:

- Double-headed black arrows represent bidirectional connections.
- Dotted green dashed arrows represent the path of example message.

Figure 46. Distributed Singleton Forwarder Network on a WAN

10.4.1 Optimizing Distributed Forwarding across a WAN

Forwarding across a WAN can be optimized by adding more forwarders. To optimize distributed forwarding across a WAN, you would create one (or more) forwarder groups on each LAN. Each group can consist of one or more RTI Forwarders and individual groups can have different numbers of group members. A forwarder cannot be assigned to more than one group. A forwarder that is not explicitly assigned to any group is considered a singleton group (that is, a group consisting of a single member). There are two options available to interconnect the groups, single-portal and load-balance.

i The simple configuration with a single forwarder at each LAN is a special case of interconnecting forwarder groups in which each LAN has a singleton group.

Single-Portal Interconnections

Using single-portal interconnections, one forwarder in each group is designated as the 'portal' forwarder. The portal forwarder acts as the entryway into the forwarder group. However, each forwarder in the group is responsible for forwarding its messages out of the group. To achieve this behavior, all local forwarders connect to the portal forwarder in foreign forwarder groups. Each forwarder then forwards messages from the federates they service to each foreign portal, which in turn relays the messages to all the federates in the portal forwarder's local forwarder group.

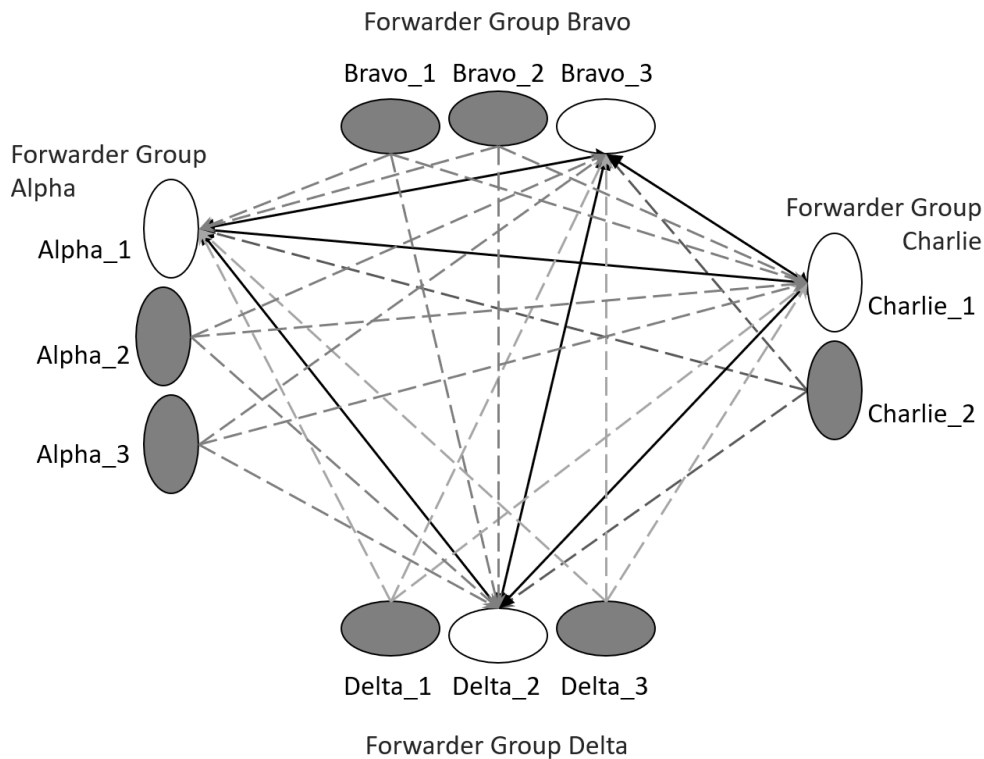
The portals also handle messages destined for the rtiexec and the distribution of internal messages to the other non-portal forwarders within their groups. Figure 47 illustrates a single-portal forwarder network. The solid lines represent bi-directional connections between two portals; the dashed lines represent connections from non-portal forwarders to portal forwarders. Non-portal forwarders send messages to the portals they are

connected to, but the portals do not send messages directly to non-portals, only to the other portals. In a forwarder group network it is the groups that are considered fully-connected, not each individual forwarder.

In the Figure 47:

- Double-headed black arrows represent bidirectional connections between portals.
- Dashed gray arrows represent non-portal forwarders to portal forwarders.
- White ellipses represent portals.
- Gray ellipses represent non-portals.

Figure 47. Distributed forwarder network (single-portal)



From a federate's perspective, the single-portal WAN configuration is similar to the LAN-based group forwarder strategy. The sending federate passes the message to its primary forwarder, which relays the message to all other federates in its forwarder group (also known as its federation subset). The RTI Forwarder also sends a copy of the message to a portal forwarder in each foreign forwarder group whose federation subsets contain federates that require the message. Those portal forwarders distribute the message to all interested federates in their group.

Load-Balanced Interconnections

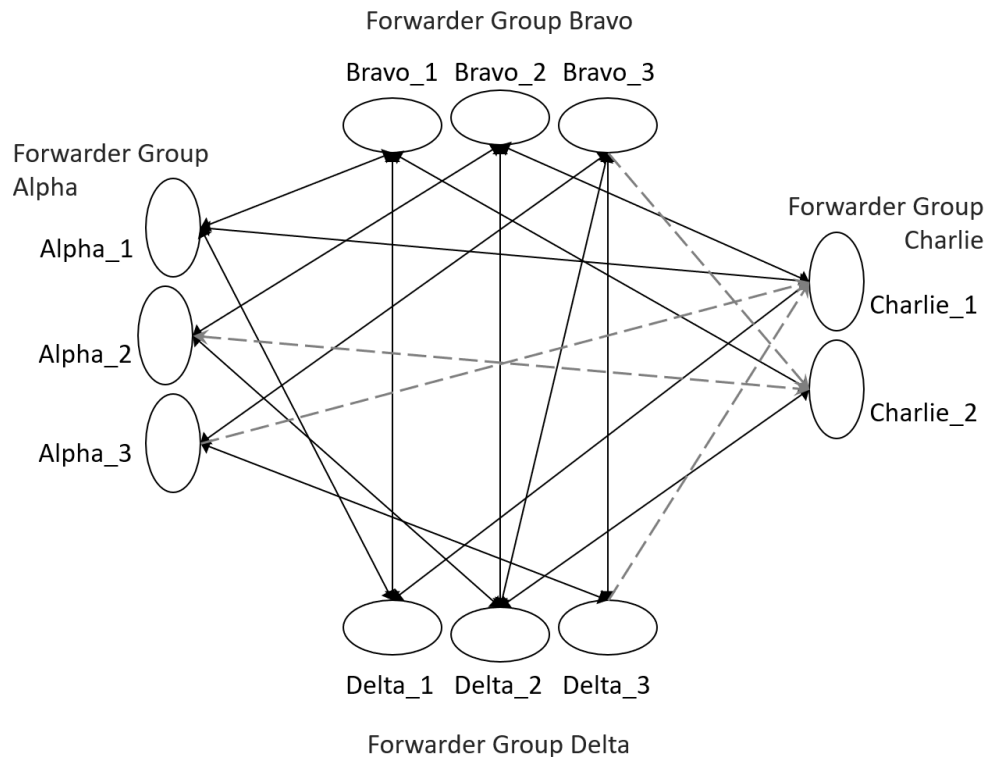
In a load-balanced forwarder network, multiple forwarders (preferably all forwarders) in each group act as portals. There are multiple entry points into the group. Again however, each forwarder is responsible for forwarding messages out of the group. Therefore, instead of all local forwarders connecting to just a single portal forwarder in foreign groups, each local forwarder selects a different portal forwarder in each foreign group. In this arrangement, each local forwarder sends messages from the federates it services to a

portal in each foreign forwarder group which in turn relays the messages to all the federates in its group. Use of multiple portals distributes the load of traffic sent between groups among several forwarders in each group, thus eliminating the potential bottleneck created by the single-portal scheme. Figure 48 illustrates a load-balanced configuration. In an ideal configuration, all groups have an equal number of forwarders and each forwarder has a bidirectional connection with a single forwarder in each foreign group.

In the Figure 48:

- Double-headed black arrows represent bidirectional connections between portals.
- Dashed gray arrows represent ancillary forwarders to portal forwarders.
- White ellipses represent portals.

Figure 48. Distributed Forwarder Network (load-balanced)



In Figure 48, there is an unequal number of forwarders in one group. One or more forwarders in this smaller forwarder group must maintain connections with more than one forwarder in the larger foreign forwarder groups. However, a forwarder with multiple connections can have only one bidirectional connection to each foreign group. The other connections to a foreign group are treated like ancillary connections, from which it only receives messages. It sends messages only to the bidirectional connection, but receives them from both. The configuration of a forwarder forming multiple connections to a single foreign group is illustrated by forwarder group Charlie, which has only two members (compared to three for the other groups). The ancillary connection from each of the other groups is automatically distributed evenly between the two portals in the group so as to avoid overloading one of the portals.

10.5. Configuring Federates for Distributed Forwarding

You configure federates for distributed forwarding by editing the RID file on each computer on which federates are running.

► **To configure federates to use distributed forwarding:**

1. Enable reliable transport, as described in "9.1. Choosing Best Effort or Reliable Transport" on page 122.
2. Specify the primary forwarder. In the RID file, set the **RTI_tcpForwarderAddr** parameter to point to the RTI Forwarder to use as the primary forwarder.
3. If you are using forwarding groups, specify the forwarders in the forwarding group. Add an **RTI-addForwarderGroupAddrString** parameter to the RID file for each RTI Forwarder in the forwarder group. If your network does not use forwarder groups, do not include this parameter in the RID file.
4. Set **RTI_distributedUdpForwarderMode** to 0, which disables distributed UDP forwarding.

i When you use forwarder groups, each forwarder serves as a primary for a subset of the federates. As a result, the configuration will be different for federates with different primary forwarders.

10.6. Configuring RTI Forwarders for Distributed Forwarding

If you use a fully-connected forwarder network across a WAN, you can configure RTI Forwarders through the RTI Assistant. For other network topologies, RTI Forwarders are configured in the routes file (*routes.mtl*). You must copy the routes file to each computer that is running an RTI Forwarder. For a given distributed forwarding topology, the routes file is the same whether it is used for a LAN or WAN.

The RTI Forwarder determines how to function:

- If you specify its connection using the RTI Assistant, it uses the selected connection configuration. It switches between distributed forwarding and centralized TCP forwarding dynamically as other forwarders connect to and disconnect from the network. It ignores the routes file even if it exists.
- If you specify use of the connection configured in the RID file by specifying the `--manual` command-line option, or by setting **RTI_configureConnectionWithRid** to 1:
 - If there is a routes file, the RTI Forwarder uses distributed forwarding.
 - If there is no routes file, it uses centralized TCP forwarding.

10.6.1 Configuring RTI Forwarders Using the RTI Assistant

To configure multiple forwarders using the RTI Assistant, you specify the network address of one of the forwarders in the Advanced Forwarder Options group box of the connection configuration dialog box. As you add RTI Forwarders, they dynamically share their network addresses to make connections.

► To configure distributed forwarder connections:

1. When creating or editing a connection configuration, click the Advanced button. The Add New RTI Connection dialog box expands (Figure 49).

Figure 49. Expanded Add New RTI Connection dialog box

The screenshot shows the 'Add New RTI Connection' dialog box. The 'Advanced' button is highlighted, and the 'Advanced Forwarder Options' section is expanded. The fields are as follows:

- Connection Name: DEV-705 1
- Host Name: DEV-705
- ☒ Use rtiexec/Forwarder
- TCP Address: 127.0.0.1
- TCP Port: 4001
- UDP Destination Address: 127.555.555.555
- UDP Port: 4000
- UDP Interface Address: 127.0.0.1
- ☒ Use IPv4 ☐ Use IPv6
- Advanced Forwarder Options:
 - Forwarder Port: 5000
 - ☐ Specify Additional Forwarder Connection
 - Address: (empty field)

2. Optionally, specify a port for the RTI Forwarder.
3. Select Specify Additional Forwarder Connection.
4. Enter the address of the RTI Forwarder that you want this forwarder to connect to. (It can be any forwarder on the network.)
5. Click OK.

10.6.2 Creating a Routes File

The routes file must specify each RTI Forwarder that will be part of the network (using **RTI_addForwarder**). The **RTI_distributedForwarderPort** parameter, in *rid.mtl*, specifies the port on which each RTI Forwarder will accept incoming connection attempts from other forwarders.

If you do not use forwarder groups, these are the only parameters required to construct a fully-connected forwarder network. A fully-connected network is one in which each forwarder maintains a connection to every other forwarder in the network, as illustrated in Figure 47 on page 144.

The list of forwarders should be identical in each routes file. The name:IP address pairs must be identical (that is, for a given IP address, the associated name must always be the same).

The syntax of the **RTI_addForwarder** parameter is:

```
(RTI_addForwarder "ForwarderName" "IP address | hostname" Port)
```

where:

- *ForwarderName* is an arbitrary name assigned to a forwarder. It cannot contain spaces.
- *IP address* is the IP address of the computer on which the forwarder is running.
- *hostname* is the host name of the computer on which the forwarder is running.
- *Port* is the port to be used to connect to this forwarder. If you specify 0, the value of the **RTI_distributedForwarderPort** RID parameter is used. The only situation where a value other than 0 would normally be specified would be when trying to run two forwarders on the same computer.

Both input values must be enclosed by double quotes (" ").

 The use of hostnames rather than IP addresses in the routes file is strongly discouraged. Although specifying hostnames may work in some limited cases, in most WAN situations, issues such as network address translation (NAT) traversal or VPN tunneling will lead to incorrect or unstable routing networks.

This example is a routes file with three forwarders:

```
;; Forwarder List
;; List all Distributed Forwarder endpoint IPs here
(RTI_addForwarder "A" "192.168.1.1" 0)
(RTI_addForwarder "B" "192.168.1.2" 0)
(RTI_addForwarder "C" "192.168.1.3" 0)
```

Table 8 describes all of the routes file parameters.

Table 8. Routes file parameters

Parameter	Description
RTI_addForwarder	Specifies the IP address (or hostname) of an RTI Forwarder to be added to the forwarder list.
RTI-enableCompression	Enables compression and sets the compression level for individual forwarder connections.
RTI-enableBundling	Enables bundling and sets the bundle size for individual forwarder connections.
RTI_forwarderInterfaceAddr	Specifies the IP address of the host interface on which to try to initiate connections to other RTI Forwarders. Only necessary for multi-homed hosts.

Table 8. Routes file parameters (continued)

Parameter	Description
RTI-allowMulticastForwarding	Specifies the forwarder name of an RTI Forwarder enabled to handle distributed UDP forwarding duties on its LAN. A separate entry for each RTI Forwarder that will act as the distributed UDP forwarder for its LAN is necessary. It is required only when forwarder groups are used and RTI_distributedUdpForwarderMode is 2.
RTI_forwarderGroupInterconnectMethod	Specifies the type of distributed forwarder network that will be created. Acceptable values are: "single-portal" "load-balance" "manual" For information about single-portal and load-balanced networks, see "10.6.4 Configuring Single-Portal Forwarder Groups" on the next page and "10.6.5 Configuring Load-Balanced Forwarder Groups" on page 151. For information about manual configuration of forwarder networks, see "10.9. Special Distributed Forwarder Networks" on page 155.
RTI-addForwarderGroup	Specifies the name of a Forwarder Group that will be included in the distributed forwarder network.
RTI-addForwarderToGroup	Adds the named RTI Forwarder to the specified Forwarder Group.

Routes File Search Order

The search order for the routes file is:

1. Load *routes.mtl* from the working directory.
2. Try to load the routes file from the directory specified by the **RTI_CONFIG** environment variable.
3. Use the default settings.

You can specify a different name for the routes file with the **RTI_forwarderRoutesFile** parameter.

10.6.3 UDP Forwarding

If you use distributed UDP forwarding to handle multicast traffic between LANs, you must add an **RTI-allowMulticastForwarding** parameter for the RTI Forwarder on each LAN that will monitor and relay those messages. The syntax is:

```
(RTI-allowMulticastForwarding "ForwarderName")
```

The *ForwarderName* string cannot contain spaces and must be enclosed by double quotes (" "). Each name must be identical to a name specified in one of the **RTI_addForwarder** parameters.

10.6.4 Configuring Single-Portal Forwarder Groups

If you are using forwarder groups to manage network traffic, you must configure the group membership rosters and the interconnection between groups in the forwarder network.

The **RTI_forwarderGroupInterconnectMethod** parameter specifies how the RTI Forwarders will assemble themselves into a network. The syntax of the parameter is:

```
(setqb RTI_forwarderGroupInterconnectMethod {"single-portal" |  
"load-balance" | "manual"})
```

where:

- *single-portal* specifies a single portal interconnection method, as described in "Single-Portal Interconnections" on page 143. This is the default interconnect method.
- *load-balance* specifies a load balanced interconnection method, as described in "Load-Balanced Interconnections" on page 144.
- *manual* specifies that interconnections are configured manually as described in "10.9. Special Distributed Forwarder Networks" on page 155.

The value string must be enclosed in double-quotes (" ").

To configure the group membership rosters, create a group using the **RTI-addForwarderGroup** parameter. Then populate it with RTI Forwarders using the **RTI-addForwarderToGroup** parameter. The RTI Forwarders must have already been specified in the forwarder list created by the **RTI_addForwarder** entries. The syntax of the two parameters is:

```
(RTI-addForwarderGroup "GroupName")  
(RTI-addForwarderToGroup "GroupName" "ForwarderName"  
"URL" acceptRemoteConnections)
```

where:

- *ForwarderName* must be identical to a name specified in one of the **RTI_addForwarder** parameters and it must appear before the group configuration section of the routes file.
- *URL* is currently not supported and should be left as an empty string (" ").
- *acceptRemoteConnections* specifies that the forwarder is a portal (1) or not a portal (0).

The *GroupName* and *ForwarderName* strings cannot contain spaces and must be enclosed by double-quotes (" ").

If none of the forwarders in a group is explicitly assigned to act as the portal for the group, the forwarder in the group whose name occurs first alphabetically acts as the portal.

The following example configuration shows the creation of four groups interconnected using the single-portal method (illustrated in Figure 47 on page 144):

```
;; Forwarder Groups
(RTI-addForwarderGroup "Alpha")
(RTI-addForwarderToGroup "Alpha" "Alpha_1" "" 1)
(RTI-addForwarderToGroup "Alpha" "Alpha_2" "" 0)
(RTI-addForwarderToGroup "Alpha" "Alpha_3" "" 0)

(RTI-addForwarderGroup "Bravo")
(RTI-addForwarderToGroup "Bravo" "Bravo_1" "" 0)
(RTI-addForwarderToGroup "Bravo" "Bravo_2" "" 0)
(RTI-addForwarderToGroup "Bravo" "Bravo_3" "" 1)

(RTI-addForwarderGroup "Charlie")
(RTI-addForwarderToGroup "Charlie" "Charlie_2" "" 0)
(RTI-addForwarderToGroup "Charlie" "Charlie_1" "" 0)

(RTI-addForwarderGroup "Delta")
(RTI-addForwarderToGroup "Delta" "Delta_1" "" 0)
(RTI-addForwarderToGroup "Delta" "Delta_2" "" 1)
(RTI-addForwarderToGroup "Delta" "Delta_3" "" 0)

(setqb RTI_forwarderGroupInterconnectMethod "single-portal")
```

There is no portal explicitly identified in group Charlie. Charlie_1 is chosen by default to be the portal because it is first in its group alphabetically (even though it is listed last in the roster for the group.)

10.6.5 Configuring Load-Balanced Forwarder Groups

To optimize load balancing of network traffic, the forwarder groups must be configured such that each of the forwarders in the group is a portal. Figure 48 on page 145 depicts such a network. The configuration of that example is given below:

```
;; Forwarder Groups
(RTI-addForwarderGroup "Alpha")
(RTI-addForwarderToGroup "Alpha" "Alpha_1" "" 1)
(RTI-addForwarderToGroup "Alpha" "Alpha_2" "" 1)
(RTI-addForwarderToGroup "Alpha" "Alpha_3" "" 1)

(RTI-addForwarderGroup "Bravo")
(RTI-addForwarderToGroup "Bravo" "Bravo_1" "" 1)
(RTI-addForwarderToGroup "Bravo" "Bravo_2" "" 1)
(RTI-addForwarderToGroup "Bravo" "Bravo_3" "" 1)

(RTI-addForwarderGroup "Charlie")
(RTI-addForwarderToGroup "Charlie" "Charlie_2" "" 1)
```

```
(RTI-addForwarderToGroup"Charlie" "Charlie_1" "" 1)

(RTI-addForwarderGroup "Delta")
(RTI-addForwarderToGroup"Delta" "Delta_1" "" 1)
(RTI-addForwarderToGroup"Delta" "Delta_2" "" 1)
(RTI-addForwarderToGroup"Delta" "Delta_3" "" 1)

(setqb RTI_forwarderGroupInterconnectMethod "load-balance")
```

The RTI Forwarders automatically determine the necessary connections for each individual forwarder and assemble themselves into the optimal forwarder network.

If it is not feasible to have every forwarder in a group act as a portal (for instance if some forwarders in a group are behind a firewall and no port forwarding is available) they should be marked as non-portals (`acceptRemoteConnections = 0`) in their

RTI-addForwarderToGroup entry. The other RTI Forwarders will automatically distribute connections to the affected group among the portals in that group. The non-portal forwarders will still connect to other portals in foreign forwarder groups, and these connections are accounted for when determining the optimal interconnect routing to ensure that every portal has a roughly equal number of connections to foreign forwarder groups to handle.

10.6.6 Configuring Distributed UDP Forwarding

The RTI Forwarder can send both TCP and UDP messages across a WAN. (UDP messages are sent between forwarders over a TCP connection.). The federates on a LAN send messages to the multicast address on their LAN. When you enable distributed forwarding for UDP messages, a single RTI Forwarder running on the LAN monitors the multicast messages and sends them to all the remote forwarder groups using the TCP network. The portal forwarder in that remote group then multicasts the messages to federates on its local LAN.

► **To configure distributed UDP forwarding:**

1. Set the RID parameter **RTI_distributedUdpForwarderMode** to 2.
2. Configure distributed forwarding as you would for TCP.
3. Specify a single forwarder on each LAN that will be allowed to monitor multicast traffic by adding **RTI-allowMulticastForwarding** entries in the routes configuration files.

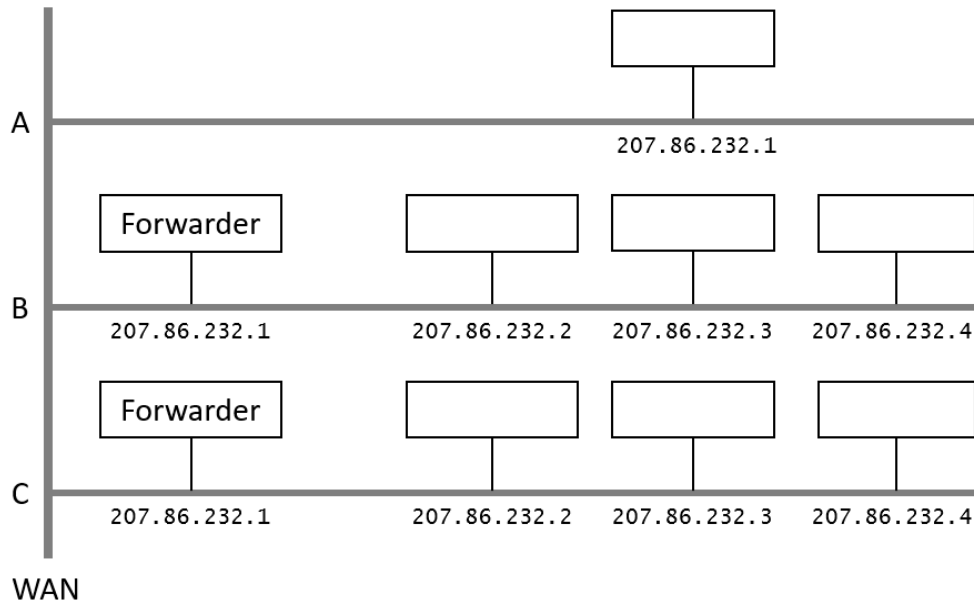
Sending Packets Directly to Remote RTI Forwarders Using UDP Unicast

An alternative way of supporting distributed UDP forwarding is to bypass the local RTI Forwarder and have each federate send a message directly to all of the remote RTI Forwarders via UDP unicast. To use this strategy, set the **RTI_distributedUdpForwarderMode** parameter to 1.

! You must use distributed forwarding to communicate with multiple federates on a single remote host that cannot be reached via multicast. Adding an additional destination address to the host is not sufficient, because unicast UDP packets are only available to one application on a single IP:port pair. This UDP unicast limitation also means that the RTI Forwarder must run on a dedicated host so that it is assured of receiving any remote messages.

Assume LANs A, B, and C (Figure 50). LAN A has one federate. LANs B and C have multiple federates and each have a RTI Forwarder. Each federate's RID file must list the multicast address for the local LAN (**RTI_destAddrString**) and the address of each remote forwarder (**RTI-addDestAddrString**) (or machine for LANs that do not run a forwarder).

Figure 50. UDP forwarding



For the federate on LAN A, there are no other local federates; its only destinations are the federates on LANs B and C. Even so, there are some messages that can be sent by an LRC that are processed by the sending LRC (for example, `requestAttributeValueUpdate`). Therefore, the primary destination address must still be either a broadcast or multicast address (which will loop back the message). Then the addresses of the two RTI Forwarders at LAN B and C would be added as additional destinations.

The RID file for LAN A might have the following entries:

```
(setqb RTI_destAddrString "229.7.7.7")
(RTI-addDestAddrString "208.86.232.1")
(RTI-addDestAddrString "209.86.232.1")
```

The RID files for federates on LAN B should be configured with a multicast address as the primary destination for the other federates on the LAN. Then the address of the federate on LAN A and the UDP Forwarder on LAN C should be added as additional destinations.

```
(setqb RTI_destAddrString "229.7.7.7")
(RTI-addDestAddrString "207.86.232.1")
(RTI-addDestAddrString "209.86.232.1")
```

The RID files for federates on LAN C should be configured similarly, but use LAN B's UDP Forwarder address.

```
(setqb RTI_destAddrString "229.7.7.7")
(RTI-addDestAddrString "207.86.232.1")
(RTI-addDestAddrString "208.86.232.1")
```

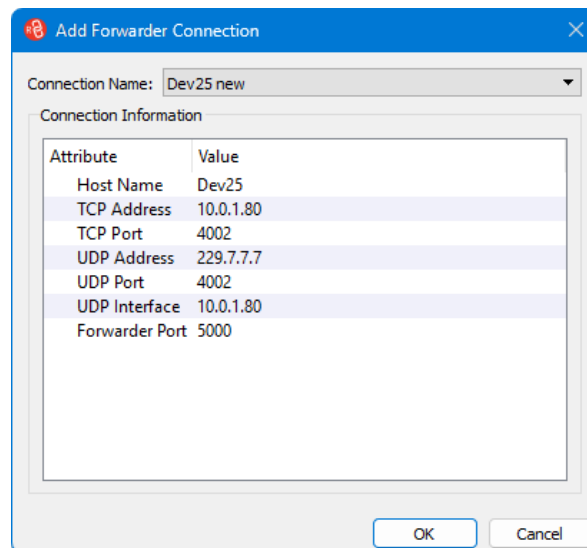
10.7. Connecting to Remote Forwarders using the RTI Assistant

You can use the Federations View to connect forwarders whose RTI Assistants are connected into a fully connected forwarder mesh network. This allows the Forwarder network to be changed dynamically, even after the rtiexec starts.

► To connect to another forwarder:

1. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
2. Choose **Federations View**.
3. In the Current Connection list, select the connection that you want to add a forwarder connection to.
4. Choose **Forwarder > Connect to Forwarder**. (If there are no connections available, this menu option is unavailable.) The Add Forwarder Connection dialog box opens.

Figure 51. Add Forwarder Connection dialog box



5. In the Connection Name list, select the connection you want to connect to.
6. Click OK. The forwarder is added to the network map.

10.7.1 Connecting to Remote Forwarders on the Command Line

Forwarders can connect to remote Forwarders via the command line. The `--forwarderToConnectTo` command-line option allows users to specify the IP address of a remote Forwarder, which will automatically result in a fully connected forwarder mesh as though the Forwarders were connected through the RTI Assistant. See "5.3. The RTI Forwarder Application" on page 74 to learn about the Forwarder command-line arguments.

10.8. Configuring Routers or Firewalls for WAN Federations

As explained in "Configuring Routers or Firewalls to Support WAN-based Federations" on page 137, when you use distributed forwarding, you will probably need to configure your system to allow traffic to get through firewalls to your RTI Forwarders.

If you are using distributed forwarding, you will typically be running one copy of the RTI Forwarder application on each LAN. But if you tried to configure your *routes.mtl* file with the IP addresses of the various machines that are running the RTI Forwarders (as described in "10.6. Configuring RTI Forwarders for Distributed Forwarding" on page 146), those addresses would be unreachable, since those machines are behind firewalls.

► **To configure distributed forwarders:**

1. Configure your *routes.mtl* file so that it uses the IP addresses of the various LANs' firewalls.
2. Configure your firewalls so that they will forward TCP packets received on the RTI's distributed forwarder port to the machines that host the RTI Forwarders. The port number used for messages destined for distributed RTI Forwarders is indicated by the **RTI_distributedForwarderPort** parameter in *rid.mtl*.

10.9. Special Distributed Forwarder Networks

The distributed forwarder networks described in previous sections can all be described as fully-connected in that each singleton forwarder or forwarder group connects to all other singleton forwarders or groups. Where forwarder groups are used, each group connects to every other group. The appropriate connections between individual forwarders are determined automatically at run-time when single-portal or load-balanced interconnects are selected in the routes file.

However, certain situations may dictate more complex network layouts. Manual configuration of forwarder interconnections permits you to construct an intricate forwarder network.

Such situations are likely to involve forwarders that cannot communicate with all other forwarders to produce a fully-connected network, whether because of infrastructure restrictions or security concerns. In place of a fully-connected network, you can set up a hierarchical network.

10.9.1 Configuring Hierarchical Forwarder Networks

To create a hierarchical forwarder network, every connection between individual forwarders must be specified in the *routes.mtl* file and the routing of messages to and from adjacent nodes must be specified for each forwarder.

i Large forwarder networks can be quite complex. If you plan to configure a hierarchical network, we recommend that you contact MAK technical support for assistance.

To set up a hierarchical forwarder network, set the **RTI_forwarderGroupInterconnectMethod** parameter to manual.

Specify connections between adjacent forwarder nodes by including an **RTI_addForwarderConnection** entry in the routes file for each connection. The syntax of this parameter is:

```
(RTI_addForwarderConnection "ForwarderName1" "ForwarderName2")
```

The *ForwarderName* strings cannot contain spaces and must be enclosed by double-quotes (" "). The names must be identical to a name specified in one of the **RTI_addForwarder** parameters. The **RTI_addForwarder** parameters must be specified before the connections between them are specified.

The routing of messages through a forwarder node is specified by including an **RTI_addForwarderRoute** entry in the routes file for each adjacent node. The syntax of this parameter is:

```
(RTI_addForwarderRoute "LinkNode" "Source" "NextHop")
```

where:

- *LinkNode* is the RTI Forwarder through which the message passes.
- *Source* is the RTI Forwarder from which the message originates.
- *NextHop* is the destination RTI Forwarder for this network hop.

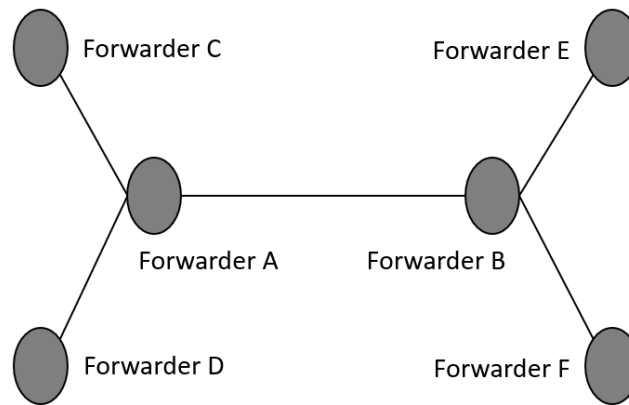
The *LinkNode*, *Source*, and *NextHop* strings must be valid forwarder names. There can be multiple entries between source and link nodes with different next hop nodes, and likewise there can be multiple entries dictating different sources through a link node to any particular next hop node.

Each of these entries configures a routing rule at the link node such that any message received by the link node from the source node will be forwarded to all the next hop nodes for which a matching routing rule exists at the link node. It is not necessary to establish a route beyond the next hop from any link node. However forwarder routes are uni-directional, so to establish a complementary return path for messages, a separate **RTI_addForwarderRoute** entry with the source and next hop values swapped is required.

Consider the simple hierarchical network illustrated in Figure 52. Forwarders A and B serve as link nodes between all other forwarders. Consequently routes must be configured to direct messages from Forwarder C through Forwarder A to Forwarders B and D, and both reverse routes must be configured (from B through A to C and from D through A to C). Likewise routes from Forwarder D through Forwarder A to Forwarder C and B (and the reverse) must be created, as well as routes from Forwarders E and F through Forwarder B and their reverse routes. No explicit entry is necessary to spell out the path from Forwarder C to Forwarder E since the entries covering the path from C through A to

B and from A through B to E will allow the messages to be delivered end-to-end. Since in this example Forwarders C, D, E, and F are leaf nodes, none of the routes go through them. In other words, leaf nodes cannot be link nodes.

Figure 52. Hierarchical Forwarder Network



The necessary entries in the routes file to create this simple network are:

```

(RTI_addForwarderConnection "A" "B")
(RTI_addForwarderConnection "A" "C")
(RTI_addForwarderConnection "A" "D")
(RTI_addForwarderConnection "B" "E")
(RTI_addForwarderConnection "B" "F")

(RTI_addForwarderRoute "A" "C" "B")
(RTI_addForwarderRoute "A" "B" "C")

(RTI_addForwarderRoute "A" "C" "D")
(RTI_addForwarderRoute "A" "D" "C")

(RTI_addForwarderRoute "A" "D" "B")
(RTI_addForwarderRoute "A" "B" "D")

(RTI_addForwarderRoute "B" "E" "A")
(RTI_addForwarderRoute "B" "A" "E")

(RTI_addForwarderRoute "B" "E" "F")
(RTI_addForwarderRoute "B" "F" "E")

(RTI_addForwarderRoute "B" "F" "A")
(RTI_addForwarderRoute "B" "A" "F")

```




11. Configuring HLA Services

For a listing of all configuration parameters, with brief descriptions, see "Appendix A. RTI Settings Reference" on page 203.

11.1. Enabling and Disabling HLA Services	160
11.1.1 Configuring MOM Services	160
11.1.2 Configuring Time Management Services	162
11.2. Configuring Internal Messages	163
11.2.1 Configuring the Delay Time for Processing Internal Messages	163
11.2.2 Transmitting Internal Messages Across Ticks	164

11.1. Enabling and Disabling HLA Services

MOM, time management, and DDM services may incur additional overhead in the RTI. Since many federations do not use these services, or only some of them, the default MAK RTI configuration disables them to optimize RTI performance. If your federation requires full HLA compliance, you can enable all of these services by setting **RTI_forceFullCompliance** to 1. If your federation uses a subset of the HLA services, you can enable or disable them independently.

11.1.1 Configuring MOM Services

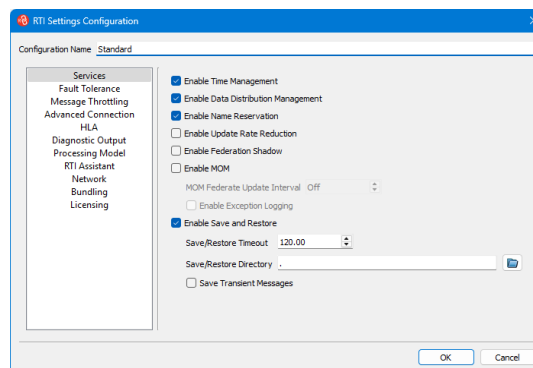
You can enable and disable MOM services and generate and log exception reports.

Enabling MOM Services

► To enable MOM services:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 53).

Figure 53. Services page



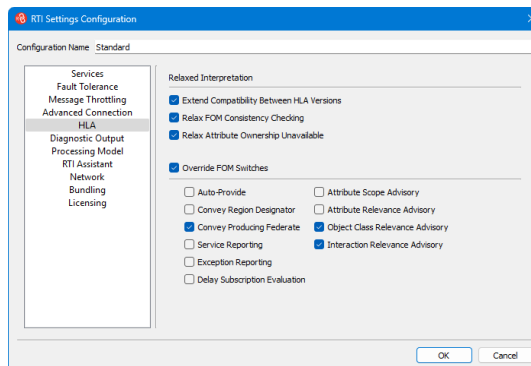
8. Select the Enable MOM check box.
9. Click OK.

Generating and Logging MOM Exceptions

► To generate and log MOM exceptions:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the HLA page (Figure 54).

Figure 54. HLA page



8. Select the Override FOM Switches check box.
9. Select the Exception Reporting check box.
10. Select the Services page (Figure 53 on the previous page).
11. Select the Enable MOM check box.
12. Select the Enable Exception Logging check box.
13. Click OK.

Setting the MOM Federate Update Interval

► To set the MOM federate update interval:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.

4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 53 on page 160).
8. Select the Enable MOM check box.
9. Select a value in the MOM Federate Update Interval box.
10. Click OK.

Enabling MOM Service Reports

You can instruct the RTI to enable MOM service reports even if the federate does not enable them using the RTI Ambassador service call. This feature may be useful for managing federations that have federates that do not automatically enable service reports.

► To enable MOM service reports:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the HLA page (Figure 54 on the previous page).
8. Select the Override FOM Switches check box.
9. Select the Service Reporting check box.
10. Click OK.

11.1.2 Configuring Time Management Services

Time management requires the use of the rtiexec.

► To enable time management:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.

4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 53 on page 160).
8. Select the Enable Time Management check box.
9. Click OK.

11.2. Configuring Internal Messages

The configuration options for internal messages help you manage internal network traffic. The implementation of the HLA services requires messages to be exchanged between the RTI components. To adhere to the HLA Interface Specification, these message exchanges can sometimes be redundant and also cause message bursts. A slight relaxation of the specification allows the RTI to regulate these internal messages in order to improve performance. You can use the configuration options in this section to regulate the internal message traffic.

11.2.1 Configuring the Delay Time for Processing Internal Messages

The RTI can delay the processing of requests by enabling the Enable Message Throttling RTI setting.

To configure the delay, set the Transmit Delay setting to the amount of time (in seconds) that the RTI should delay before processing requests.

During federation startup it is typical for federates to perform declaration and object management services that require the RTI to exchange a significant number of internal messages. Often, these messages are redundant, simply satisfying duplicate requests from each joining federate. For example, the LRC of a federate that has registered an object will resend the register information for each subscription message that it receives (that matches the registered object). The LRC can condense these duplicate responses (for example, register messages) by delaying the processing of requests (for example, subscription messages) allowing a single response to satisfy multiple requests. Another example is the processing of request attribute values. By delaying the invocation of the federate ambassador provide attribute value service, a single invocation can handle the multiple requests typically occurring during federation startup.

 In the 1516 APIs, the user supplied tag is dropped.

► To configure the delay time for processing internal messages:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.

3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Message Throttling page (Figure 55 on the facing page).
8. Select the Enable Message Throttling check box.
9. Enter a value in the Transmit Delay box.
10. Click OK.

11.2.2 Transmitting Internal Messages Across Ticks

The RTI can send out internal messages at a throttled rate. This is enabled by enabling the Enable Message Throttling RTI setting. To configure the transmission rate, set Transmit Rate to the rate, in messages per second, that the LRC should transmit all required internal messages across multiple ticks (based on the time since the last tick()). The default setting is 0.0, which means the LRC should send all messages during a single tick.

During federation startup it is typical for federates to perform declaration and object management services that require the RTI to exchange a significant number of internal messages. Often, the LRC has to respond to messages that it receives by retransmitting messages about its internal state (for example, object registrations and class publications). The LRC retransmits all the necessary messages during a single tick. This message retransmission process can cause a spike in the message bandwidth. To smooth out these message spikes, the LRC can be throttled to retransmit messages at a specified rate across subsequent ticks.

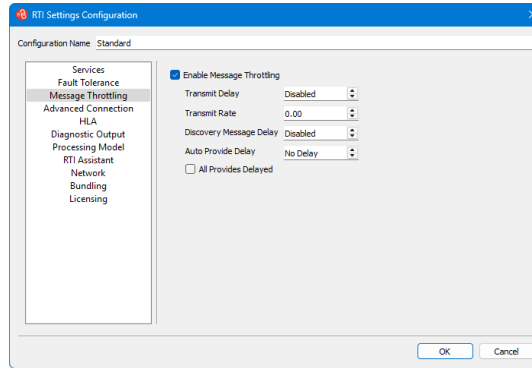
 The throttle only affects the retransmission of internal messages during tick(). It does not affect messages required by services invoked by the federate (for example, updates and interactions).

► **To configure the delay time for processing internal messages:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.

7. Select the Message Throttling page (Figure 55).

Figure 55. Message Throttling page



8. Select the Enable Message Throttling check box.
9. Enter a value in the Transmit Rate box.
10. Click OK.



12. Using Data Distribution Management

This chapter describes the approaches to Data Distribution Management (DDM) that are implemented by the MAK RTI.

12.1. Configuring DDM Services	168
12.1.1 Enabling DDM	168
12.1.2 Configuring Distributed Region DDM	169
12.1.3 Configuring a Fixed Grid DDM	169
12.1.4 Configuring Implicit DDM	171
12.2. The Distributed Region Approach	173
12.2.1 Attribute Passelization	174
12.3. The Fixed Grid Approach	175
12.3.1 The Fixed Grid Representation	176
12.3.2 Federate Region Intersections with the Fixed Grid	177
12.3.3 Normalization of Federate Data for Range Bounds	180
12.3.4 The Fixed Grid Approach and DDM Services	180
12.4. Implicit DDM	182
12.4.1 Implicit DDM Implementation	183
12.4.2 Adding Implicit DDM to the FED or FDD	183
12.4.3 Implicitly Associating DDM Regions	184
12.4.4 Modifying Region Extents	185
12.4.5 Modifying Fixed Grid Region Extents	186

12.1. Configuring DDM Services

The MAK RTI implements these approaches to DDM:

- Distributed region.
- Fixed grid.
- Implicit DDM.

This section explains how to enable DDM and configure the different DDM approaches. The rest of this chapter provides conceptual background for the different approaches to DDM.

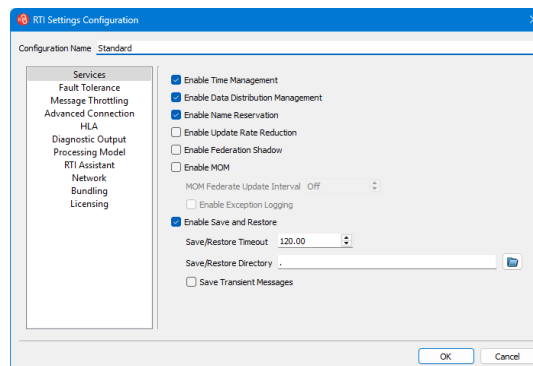
i The HLA Evolved implementation of DDM is the same as the HLA 1516 implementation. Therefore, in this chapter, all references to HLA 1516 apply to HLA Evolved as well.

12.1.1 Enabling DDM

► To enable DDM services:

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 56).

Figure 56. Services page



8. Select the Enable Data Distribution Management check box.
9. Click OK.

12.1.2 Configuring Distributed Region DDM

Distributed region DDM is explained in "12.2. The Distributed Region Approach" on page 173. To use the distributed region approach, specify values for the **RTI_minChannelAddr** and **RTI_maxChannelAddr** parameters. They define a pool of multicast addresses that the LRC will assign to regions used for sending interactions and updating attributes. For optimal DDM filtering, each federate should define a separate pool of multicast address for its LRC to use.

 The MAK RTI supports IPv4 and IPv6, but you cannot mix the two protocols. If your federation uses IPv4, then all IP addresses in the RTI settings and RTI Assistant must be in the IPv4 format. The same applies to use of IPv6.

The parameter **RTI_addressDelay** is used to temporarily bypass the use of an updated region's multicast address when sending an interaction or attribute update. During the delay, the region change can propagate to subscribing federates and allow any that form new region overlaps to receive the messages.

12.1.3 Configuring a Fixed Grid DDM

Fixed grid DDM is explained in "12.3. The Fixed Grid Approach" on page 175. To use the fixed grid approach, you enable settings in *rid.mtl* and create a file that specifies the grid. The fixed grid configuration specifies the partitioning of the axes and the association of the axes with FOM dimensions. It specifies a routing space that must contain all the dimensions used to configure the fixed grid. In HLA 1516, a notional global routing space is assumed, but it may be specified as *HlaGlobalSpace*. A dimension may be specified as the subspace axis. If specified, its partitioning determines the number of application spaces. The remaining dimensions used to define the fixed grid are listed and only these dimensions may appear in the specification of the application spaces. The dimensions must be a subset of the routing space dimensions.

The RID parameters for fixed grid DDM are:

- **RTI_ddmFixedGrid**. Enables or disables the fixed grid approach.
- **RTI_ddmFixedGridFilename**. Specifies the file name (and path) of the fixed grid specification file.
- **RTI_minChannelAddr**. Specifies the base multicast address used to initialize the fixed grid.

 For HLA 4, in addition to the RID parameters above, the *AllowRelaxedDDM* switch must be enabled in the FOM.

The syntax of the fixed grid specification file is:

```
;;with supspace;;
([RoutingSpaceName]
  (SubspacePartition subspaceDimensionName partitionSize
```

```
(dimName*) [defaultPartitionSize]
(
  [ (index (descriptiveName (dimName partitionSize)* )]*
)
)
)
;;without subspace;;
([RoutingSpaceName]
(
  (<dimName> [defaultPartitionSize])*
)
)
```

where:

- *RoutingSpaceName* specifies the name of the routing space. For HLA 1.3, the routing space name is required to get dimension handles. For HLA 1516, HLA Evolved, and HLA 4, the routine space name is optional. The default is an internal routing space named *HlaGlobalSpace*.
- *SubspacePartition* is a keyword. Its components are:
 - *subspaceDimensionName*. Specifies the FOM dimension.
 - *partitionSize*. Specifies the number of partitions for the subspace axis.
 - *dimName*. Specifies one or more dimension names of the remaining axes.
 - *defaultPartitionSize*. Specifies the default partition size.
- Each index line specifies partitions for the application space, as follows:
 - *index*. Specifies an application space, beginning with 0. If an application space index is omitted, its axes are partitioned using the default partition size.
 - *descriptiveName*. Used for documentation only.
 - *dimName*. Specifies how the dimensions will be partitioned for each application space. If the dimension name is omitted, its axes are partitioned using the default value partition size.
- `;;` indicates a comment.

This example fixed grid specification shows six application spaces in the subspace partition:

```
(RS1
  (SubspacePartition subspace 6 (one two) 1
    (
      (0 (level0 (one 1) (two 1)))
      (1 (level1 (one 2) (two 2)))
      (2 (level2 (one 4) (two 4)))
      (3 (level3 (one 8) (two 8)))
      (4 (level4 (one 16) (two 16)))
      (5 (level5 (one 32) (two 32)))
    )
  )
)
```

```
)  
)  
)
```

Fixed Grid Search Order

The fixed grid configuration file search order is as follows:

1. Try to load the grid file specified by **RTI_ddmFixedGridFilename** in the RTI settings file.
2. Try to load `RTI_CONFIG/grid_file_name_from_settings_file`.
3. If the configuration file is not found, the RTI throws an exception.

12.1.4 Configuring Implicit DDM

Implicit DDM is explained in "12.4. Implicit DDM" on page 182. You can configure the Implicit DDM plug-in. Configuration options include the:

- Upper and lower bounds of the dimensions.
- Update region size.
- Update region margin.
- Sensor range, which determines the subscription region size.
- Weapon range, which determines the interaction region size.

The configuration file uses MAK Technologies Lisp (MTL) format (as described in "Appendix A. RTI Settings Reference" on page 203). The name of this file is specified by the RTI settings parameter **RTI_implicitDdmParamsFile**. The syntax for this parameter is:

```
(setqb RTI_implicitDdmParamsFile "fileName")
```

where *fileName* is a quoted string containing the name of the implicit DDM configuration file.

Table 9 describes the parameters in the implicit DDM configuration file:

Table 9. Implicit DDM parameters

Parameter	Description
RTI_implicitDdmDefaultUpdateRange	Specifies the maximum range, in meters, that the object can travel between successive location updates. This value is not used when fixed grid DDM is enabled. The default update range is used when a range is not specified for the object class using RTI-setObjectUpdateRange. This value is not used when fixed grid DDM is enabled. Default: 0.0.
RTI_implicitDdmDefaultSensorRange	Specifies the maximum range, in meters, within which the object can sense another object. This value is added to the update range to determine the furthest point an object can sense another object between successive location updates. The default sensor range is used when a range is not specified for the object class using RTI-setObjectSensorRange. Default: 100.0.
RTI_implicitDdmRegionMargin	Specifies a factor that increases the update region range so that the region does not have to be modified each time the object updates its location. The update range is multiplied by this factor so that the region is not modified unless the resulting region extent does not encompass the object's actual update range. This value is not used when fixed grid DDM is enabled. Default: 5.0.
RTI_implicitDdmMinExtentLimit RTI_implicitDdmMaxExtentLimit	Specifies the range over which the location values are normalized. Defaults: -12000000.0 and 12000000.0.
RTI_implicitDdmOriginLat RTI_implicitDdmOriginLong	Specifies the relative center location used to normalize location values. Defaults: 0.0 and 0.0.
RTI-setObjectUpdateRange	Specifies an update range for an object class. Multiple instances of this parameter are permitted.
RTI-setObjectSensorRange	Specifies a sensor range for an object class. Multiple instances of this parameter are permitted.
RTI-setObjectWeaponRange	Specifies a weapon range for an object class. Multiple instances of this parameter are permitted.

This is an example of an implicit DDM configuration:

```
;; All distances in meters
(setqb RTI_implicitDdmDefaultUpdateRange 0.0)
(setqb RTI_implicitDdmDefaultSensorRange 2000.0)
(setqb RTI_implicitDdmMinExtentLimit -12000000.0)
(setqb RTI_implicitDdmMaxExtentLimit 12000000.0)
(setqb RTI_implicitDdmRegionMargin 6.0)
```

```
;; Latitude and Longitude in degrees
;;(setqb RTI_implicitDdmOriginLat 10.0)
;;(setqb RTI_implicitDdmOriginLong 60.0)

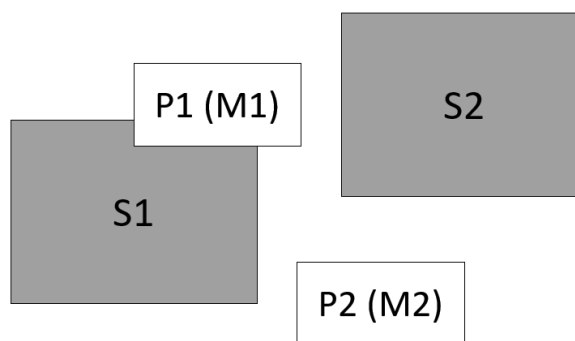
(RTI-setObjectUpdateRange "BaseEntity.PhysicalEntity.Platform.
    Aircraft" 328.8)
(RTI-setObjectSensorRange "BaseEntity.PhysicalEntity.Platform.
    Aircraft" 160000.0)
(RTI-setObjectWeaponRange "BaseEntity.PhysicalEntity.Platform.
    Aircraft" 56000.0)
```

12.2. The Distributed Region Approach

In the MAK RTI distributed region approach, each federate's LRC exchanges its region information with remote LRCs. The RTI must exchange information about update regions and subscription regions. Update regions are regions used for sending updates and interactions. Subscription regions are regions used in DDM subscriptions. The individual LRCs perform matching between the update and subscription regions. An LRC uses the region matches to establish communication channels between publishers and subscribers. Publishers then send data using the established communication channels. Only subscribers that are listening to those channels receive the data.

The MAK RTI establishes best effort communication channels by mapping the update regions to multicast groups. These mappings are also distributed among the LRCs. When an LRC's subscription region matches an update region (that is, it overlaps and has common attributes), the LRC joins the update region's multicast group. The data is sent once to the multicast address and only subscribers that have joined that multicast group receive the data. In Figure 57, each update region has its own multicast address and subscription regions join the multicast group of overlapping update regions.

Figure 57. Distributed region using update regions

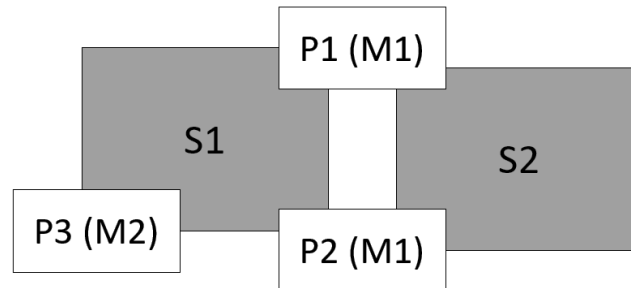


It is important to understand that multicast filtering is really just a very efficient receive side filtering. The actual bandwidth is not reduced. However, the processing at the receiver is greatly reduced as filtering occurs at the network layer rather than in the RTI layer. One caution with this approach is that the number of effective multicast groups is limited. Multicast groups will probably need to be shared among the communication channels. As a result, the RTI may receive messages that do not match the federate's subscriptions. The LRC handles these cases through an efficient channel identifier filter. Only messages with channel identifiers from matching update regions are processed.

The MAK RTI also uses region matching information for reliable transmissions. Because both update and subscription region information is exchanged between the LRCs, an LRC can determine exactly which remote region subscriptions match its local update regions. The MAK RTI uses this information to establish the destination mask used in sender-side filtering. The use of DDM matching to support smart forwarding results in true sender-side filtering. In fact, if no matching subscriptions are found on the sending side, the MAK RTI can prevent updates from being sent for both reliable and best effort messages.

In Figure 58, P1 and P2 both have a subscription set of S1 and S2 while P3 has a subscription set of S1.

Figure 58. Distributed region using subscription regions



The advantage of distributed region matching is that the LRC can establish communication channels that carry relevant data only to receiving federates (in other words, perfect filtering). Additional unsubscribed attributes may be carried through the channel, but this multiple association problem exists in practically any approach, including the fixed grid approach. Also, the transmission of attributes and interactions is always sent only to a single channel (versus fixed grid where updates may have to be sent to multiple channels).

The disadvantage of distributed region matching is the considerable overhead of distributing the region information and performing matching between regions. With the distributed region approach, the overhead required to maintain regions becomes similar to object updates (with reliable attributes). A federation that naively uses regions and region updates can easily overwhelm any advantages of DDM implemented this way. Also, region matching scales poorly with the number of regions. Therefore, federates should limit the number of regions and the frequency that the regions are updated. An analysis of how using regions affects federation performance is provided in the paper SIW 99 1-054 "Characterizing Scenarios for DDM Performance and Benchmarking RTIs".

A detailed description of the distributed region implementation for the MAK RTI is provided in 02S-SIW-056 "Implementation of DDM in the MAK RTI", which is available at <https://mak.com/docs/papers/Implementation-of-DDM-in-the-MAK-RTI-spring-SIW-2002.pdf>.

12.2.1 Attribute Passelization

When a federate invokes an `updateAttributeValues` service, the LRC separates the submitted attributes into the designated region set channels formed from previous region associations. Therefore, the LRC may transmit more than one message for a given update even though all the attributes have the same transport type and order type. The LRC

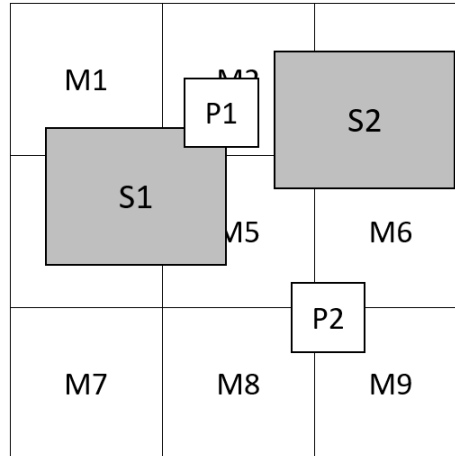
shares region sets across all object classes, but creates them separately for interaction classes. A federate should use separate regions when associating regions across multiple objects of different classes. Otherwise, the LRC will transmit the attributes from these objects using the same channel and filtering will be diminished.

12.3. The Fixed Grid Approach

A fixed grid approach to DDM allows the RTI to approximate the routing that would occur through update and subscription region overlap calculations, without having to exchange region information between LRCs. The fixed grid approach relies on an a priori configuration of multicast addresses into a fixed grid. Each axis of the fixed grid is associated with a FOM dimension. Each cell of the fixed grid is assigned a multicast address. The fixed grid takes the place of the opposing region in the DDM region overlap calculations. The federate regions are overlapped with the fixed grid and the overlapped grid cells establish the multicast addresses used for message routing. For subscription regions, the LRC joins each multicast address that is found in overlapped grid cells. For update regions, messages are sent using the multicast addresses found in overlapped grid cells.

! The fixed grid approach is not fully compliant with either HLA 1.3 or HLA 1516. It relaxes some requirements and imposes some restrictions. (For details, see "12.3.4 The Fixed Grid Approach and DDM Services" on page 180.)

Figure 59. Fixed grid approach with overlaid regions



The MAK RTI fixed grid approach is modeled after application space partitioning (introduced by Coffin, et. al. and Helfinstine, et. al. ^{1, 2}). A few generic dimensions are

¹Experimentation with DDM schemes, D. Coffin, M. Calef, D. Macanucco, and W. Civinskas, Spring 1999 Simulation Interoperability Workshop, 99S-SIW-053.

²Experiences with Data Distribution Management in Large-Scale Federations, B. Helfinstine, D. Wilbert, M. Torpey, and W. Civinskas, Fall 2001 Simulation Interoperability Workshop, 01F-SIW-032.

defined, with one "subspace" dimension used to partition the others into application spaces. The remaining dimensions can then be interpreted and partitioned to meet the specific application requirements.

The same fixed grid approach is used for both the HLA 1.3 and 1516 APIs. The application space partitioning is less useful for DDM in HLA 1516, because each attribute is associated with individual dimensions rather than routing spaces. In HLA 1.3, a region contains all the dimensions in a routing space, while in HLA 1516 a region can be comprised of exactly the dimensions needed (that is, a subset of the available dimensions). Using the same implementation makes it possible for federates that use the different APIs to interoperate.

12.3.1 The Fixed Grid Representation

The fixed grid is defined by a set of n axes that form grid cells when they are partitioned. One specific axis can be defined as a subspace axis. The partitioning of a subspace axis creates application spaces. For each application space, the partitioning of the remaining $n-1$ axes is separately defined. If no subspace axis is specified, there is one specification for the partitioning of the n axes.

Each cell of the fixed grid is assigned a multicast address. The assignment of multicast addresses starts with a base address. The addresses are assigned sequentially through a "depth first" recursion. With no subspace axis, this amounts to assigning the addresses uniformly across the dimensions. The assignment is similar with a subspace axis, but each subspace results in a distribution of addresses for the remaining dimensions based on the subspace partitioning.

Table 10 illustrates a fixed grid partitioning and multicast address assignment. The figure shows a subspace axis partitioned into six application spaces. Each application space has a single axis. In the 0th application space partition the axis has one partition, in the 1st application space it has two partitions, in the 2nd application space it has four partitions, and so on.

Table 10. Fixed grid partitions and multicast assignment

G R I D C E L L S	224.0.0.1	224.0.0.2	224.0.0.4	224.0.0.8	224.0.0.16	224.0.0.32
			224.0.0.5			
		224.0.0.3	224.0.0.6			
			224.0.0.7			
				224.0.0.15		
						224.0.0.63
Sub-space Partition	0	1	2	3	4	5

For details about configuring the fixed grid approach, see "12.1.3 Configuring a Fixed Grid DDM" on page 169.

12.3.2 Federate Region Intersections with the Fixed Grid

A federate region intersection with the fixed grid maps each dimension range onto the associated fixed grid axes. The result of this mapping is a beginning and an ending index along each of the fixed grid axes. These indices specify the overlapped grid cells that

contain the multicast addresses. The set of multicast addresses are then used for sending updates and interactions or subscribing to object class attributes or interaction classes.

Each fixed grid axis has an upper and lower bound. The lower bound is always 0. For HLA 1.3, the upper bound is the maximum value of an unsigned long integer (that is, 4,294,967,295 or ((unsigned long)~0)). For HLA 1516, the associated FOM dimension defines the upper bound. The width of a grid cell along an axis is determined by dividing the axis upper bound by the number of partitions. For an axis with n partitions the width is calculated as:

$$w = \text{upperBound}/n$$

The mapping of a region's upper or lower range bound value, v , to an index along a fixed grid axis is calculated as:

$$I = \text{floor}(v / w)$$

For example, assume the following configuration with an upper bound of 400 for all dimensions:

```
(RS1
  (SubspacePartition subspace 6 (one two) 1
    (
      (0 (level0 (one 1) (two 1)))
      (1 (level1 (one 2) (two 2)))
      (2 (level2 (one 4) (two 4)))
      (3 (level3 (one 8) (two 8)))
      (4 (level4 (one 16) (two 16)))
      (5 (level5 (one 32) (two 32)))
    )
  )
)
```

The fixed grid with the same partitioning as Table 10 would have grid cell boundaries as illustrated in Table 11. A federate region has a lower and an upper bound for each dimension in the region. Using the index formula, the lower bound is mapped to a beginning index and the upper bound is mapped to an ending index. The dimension specified as the subspace dimension is mapped first. The range for the subspace is assumed to be a point range (only the lower bound used), because a single subspace must be selected.

Given that the subspace dimension has 6 partitions, a range bound value from 0 through 66 would result an index of 0, from 67 through 133 would result in an index of 1, and so on. For the last partition, the maximum upper bound would be mapped down into the maximum index, so a value of 400 would result in an index of 5 versus 6. For the other dimensions, the index would depend on which subspace index was specified. For subspace 0, any value would result in an index of 0. For a subspace of 1, values 0 through 199 would result in an index of 0, and values of 200 through 400 would result in an index of 1. The indices for the other subspace partitions are calculated similarly.

Table 11. Grid Cell Partition Boundaries

G R I D C E L L S	399	200	300	350	375	388			
					349				
				299					
						200			
					199		100		
			99	50					
						49	25	13	
					24				
		0							
		Subspace Partition		0-66	67-133	134-199	200-266	267-333	334-399

12.3.3 Normalization of Federate Data for Range Bounds

Similar to the mapping of the region's range bounds onto the fixed grid axes, federate data must be mapped or normalized into the region range bounds. Mappings that work well with the fixed grid approach are "enumerated" (for example, vehicle type) and "linear" (for example, location).

For an enumerated mapping, the magnitude of a dimension's full extent is partitioned by the number of enumerations. The range bound is then set as a point value in the midpoint of the partition. An obvious configuration for a dimension with an enumerated mapping would be to match the fixed grid partitions on that axis with the number of enumerations. Then each value of the enumeration would specify a separate multicast address.

The formula¹ for an enumerated mapping is calculated as follows:

1. Determine a scale between the dimension extent and the enumeration:
$$\text{double scale} = \text{dimensionUpperBound} / \text{double}(\text{maxEnum} - \text{minEnum} + 1)$$
2. Move the rangeBound into the center of the partition by adding **scale/2**:
$$\text{double rangeBound} = (\text{scale} * (\text{value} - \text{minEnum})) + \text{scale} / 2$$
3. Truncate the final value to the nearest integer:
$$\text{unsigned long result} = \text{rangeBound}$$

For a linear mapping, the magnitude of the dimension's full extent is scaled by the linear extent. A typical linear mapping for a coordinate system would first define the bounds of the coordinate values (for example, minimum and maximum longitude and latitude). The partition of the fixed grid axes could then represent the resolution of filtering (that is, more partitions would allow finer-grained location filtering).

The formula for a linear mapping is similar to that for an enumerated mapping, but without the adjustment to the center of the partition. It is calculated as follows:

1. Determine a scale between the dimension's extent and the linear bounds:
$$\text{double scale} = \text{dimensionUpperBound} / \text{double}(\text{max} - \text{min})$$

$$\text{double rangeBound} = \text{scale} * (\text{value} - \text{min})$$
2. Truncate the final value to the nearest integer:
$$\text{unsigned long result} = \text{rangeBound}$$

12.3.4 The Fixed Grid Approach and DDM Services

The fixed grid approach is not fully compliant with any of the HLA specifications. It relaxes some requirements and imposes some restrictions. This section lists each of the DDM services and related object management services with a description of how the MAK RTI fixed grid DDM implementation behaves and how the services may differ from strict HLA requirements.

¹ Based on formulas used by MC02.

i In the following sections, all comments regarding HLA 1516 also apply to HLA Evolved.

Create Region

A fixed grid region is initialized with the exact dimensions that are used to configure the fixed grid. As a result, any additional dimensions are ignored (even in HLA 1.3 when they belong to the same routing space). Since the region is initialized with the fixed set of dimensions, any of these dimensions not specified when creating the region are automatically added. The initial values for the range bounds are the default range bounds. In HLA 1.3, the default range bounds are 0 to 4,294,967,295. In HLA 1516, the default range bounds are set by the FOM. If the FOM omits the default range bounds or the default range bounds are excluded, the default range bounds are 0 to 1.

Modify Region / Commit Region Modifications

A region can be committed without setting all of its range bounds. Any range bound that has not been set assumes the dimension's default range. In HLA 1516, you cannot exclude fixed grid dimensions. Excluded dimensions take on the default range bound values of 0 to 1.

Delete Region

This service behaves as expected. A region in use cannot be deleted.

Register Object Instance With Region

For a given object instance, all of its attribute instances must use one, and only one, region. The RTI uses the first region and ignores any others. Registering an object with region for any attribute results in all attributes being associated with that region.

Associate Object Instance With Region

All object attribute instances must use one, and only one, region. The RTI uses the first region and ignores any others. Associating an object with region for any attribute results in all attributes being associated with that region. Subsequent associations must be with that same region and thus are redundant, including a region used to register the object.

Unassociate Object Instance With Region

All attribute instances of the object become unassociated with region when any one or more attributes are unassociated.

Subscribe Object Class Attributes With Region

Multiple regions can be used for object class attribute subscriptions, including successive subscriptions. However, the object class attribute subscriptions are additive across all regions. In essence, an additive declaration management subscription occurs along with the subscription to the regions' multicast addresses. The result is that all subscribed attributes that match an update are reflected regardless of which region's multicast subscription transported them.

Unsubscribe Object Class Attributes With Region

All attributes of a class are unsubscribed from the region. When no regions are subscribed to the object class attributes, the object class is unsubscribed. Any declaration management subscriptions to the class are also unsubscribed. Further, a declaration management unsubscription results in those attributes no longer being included in reflects (even though the messages are received by the LRC).

Subscribe Interaction Class With Regions

An interaction subscription with region behaves as expected.

Unsubscribe Interaction Class With Regions

The region is unsubscribed from the interaction class. When no regions are subscribed with the interaction class, the interaction becomes unsubscribed. Any declaration management subscription is also unsubscribed. Further, a declaration management unsubscription results in the interaction no longer being received (even though the messages are received by the LRC).

Send Interaction With Regions

When sending interactions with regions, only point regions are supported. The RTI only uses the lower bound.

Request Attribute Value Update With Regions

A request with region is sent to each of the region's multicast addresses.

Update Attribute Values

When sending attribute updates that have been associated with regions, only point regions are supported. The RTI only uses the lower bound.

Miscellaneous

In HLA 1.3, the region's first extent is used and any additional extents are ignored. In HLA 1516, a region only has one extent.

The fixed grid approach disables all advisories. No subscription or publication information is exchanged between federate LRCs.

12.4. Implicit DDM

Implicit DDM provides the advantages of DDM filtering without a federate using the DDM services. Implicit DDM uses a MAK RTI plug-in that substitutes non-DDM method calls from a federate with DDM method calls. The DDM calls create regions and modify region extents using a configurable set of criteria, based on geographic data as specified by the RPR FOM.

The Implicit DDM plug-in provides geographic range-based filtering. It allows you to specify three ranges: an update range, a sensor range, and a weapon range, for any RPR FOM BaseEntity object, but primarily for PhysicalEntity::Platform objects. The update range is the distance an entity can move in one update interval. The sensor range is the range in which an entity can sense other vehicles. The weapon range is the range in which an entity can engage another entity. It allows different BaseEntity subclasses to be

configured with different ranges to reflect the different capabilities of the platform. For example, the sensor range of an aircraft can be set to a different value than that of submersible vessels.

The effect of the Implicit DDM plug-in is that as the federate executes, it will only receive updates and interactions from entities that are within range of the federate's entities. Based on implicitly created and modified DDM regions, entities are within range when the update range or weapon range of discovered entities overlaps the sensor range of the federate's registered entities.

The plug-in works with either the MAK RTI's distributed region approach (described in "12.2. The Distributed Region Approach" on page 173) or its fixed grid approach (described in "12.3. The Fixed Grid Approach" on page 175.) At runtime the plug-in determines what type of approach is in use based on RID file parameters and adjusts its behavior so you can use the same plug-in DLL (or so) regardless of the underlying DDM approach.

The following sections describe how implicit DDM is implemented and how to configure the RTI to use it.

12.4.1 Implicit DDM Implementation

The Implicit DDM plug-in uses the RTISpy API. (For details about the RTISpy API, see the online API documentation.) The plug-in changes the way objects are registered, published, subscribed, and updated, and introduces changes when the federation joins (for example, modifying the RTI's internal FED routing space).

i Almost all of the the plug-in's work is performed above the actual RTI Ambassador implementation, with perhaps some alterations to other classes employed by the RTI Ambassador such as the FOM Manager (*DtFomClassManager*). It was not necessary to change the Federate Ambassador to implement the implicit DDM approach.

Since the plug-in must intercept and decode attribute value updates and interactions, it needs some knowledge of the underlying federation object model. The Implicit DDM plug-in was designed to be compatible with HLA RPR FOM 1.0 and HLA RPR FOM 2.0 Draft 17 or their derivatives. All references to actual classes, attributes, interactions, and parameters will use RPR FOM 1.0 nomenclature.

12.4.2 Adding Implicit DDM to the FED or FDD

During the join process, the Implicit DDM plug-in creates its own routing space with its own dimensions representing geocentric coordinates. In HLA 1.3, the plug-in adds a routing space named "ImplicitDdmSpace" to the RTI's internal FED. In the 1516 specifications, just the dimensions are added to the RTI's internal FDD. The added dimensions are called HLA_X, HLA_Y, and HLA_Z and correspond to the geocentric coordinates X, Y, and Z. The routing space and dimensions are associated with the attributes of certain classes of objects (*BaseEntity*) and to certain interactions associated with those objects (*WeaponFire* and *MunitionDetonation*).

The implicit DDM routing space is automatically added to the RTI's internal FED and it should not be declared in the actual FED file.

The 1516 implementations of the Implicit DDM plug-in add the implicit DDM dimensions to the MAK RTI's 1516 global routing space.

Normalization and Initial Bounds

By default the implicit DDM dimensions are centered on the geocentric origin (0,0,0). This can be changed to any point on the Earth's surface by specifying the latitude and longitude of the origin in the implicit DDM configuration file.

The default implicit DDM routing space defines the limits of each dimension to be:

- min - -12000000.0 m.
- max - 12000000.0 m.

The approximate radius of the Earth is 6000 km. Therefore, the default space subtends a cube roughly twice the diameter of the Earth on all sides. The minimum and maximum extent limits are configurable.

All the dimensions are normalized to take values between 1 and MAX_INT-1 (for 1516 the normalized boundaries are from 2 to MAX_INT-2). This range allows for point regions to be created at {0,0,0} and {MAX_INT,MAX_INT,MAX_INT} which an object can never occupy.

Because the DDM is implicit, no region information is associated with an object instance when it is registered. So until the first location attribute for the new object instance is decoded by the plug-in, it cannot set the region bounds based on the actual coordinates of the object instance. Instead, the plug-in creates initialization regions and sets the initial bounds to one of the two inaccessible points. This ensures that when a new object instance is created, its update regions will not inadvertently overlap with any remote federate's subscription region and trigger erroneous detection by remote federates. Similarly, the federate's subscription region will not inadvertently overlap with remote federate update regions.

12.4.3 Implicitly Associating DDM Regions

The Implicit DDM plug-in keeps track of each instance of certain object classes. It maintains a list of handles for all the object instances of type BaseEntity that are registered by the federate. For each such instance it creates two regions: an update region (the update range) similar to that used in normal DDM processing, and an interaction region (the weapon range). The plug-in also creates a single subscription region (the sensor range) used by all objects. For each registered object it adds a subscription extent.

 HLA 1516 APIs do not support subscription regions with multiple extents. However it was necessary for the plug-in to be able to create and modify an arbitrary number of extents within each object's subscription region. Consequently, in certain circumstances, the range-based Implicit DDM plug-in bypasses the HLA 1516 API calls to directly access the underlying classes.

The plug-in then intercepts the attribute updates and interactions from each instance of the desired classes and extracts location information. Before it allows the attribute updates or interactions to be processed further, the plug-in uses the location information to modify the extents of the regions it has created and to make the necessary DDM calls to distribute the region modifications. If necessary, a DDM call is substituted for the normal

DM call (for example, `sendInteractionWithRegion`). No additional changes are necessary, because the normal underlying DDM processing uses the implicit DDM region information as if the federate had made the DDM calls directly.

12.4.4 Modifying Region Extents

Region extents are modified slightly differently depending on the DDM approach that is being used by the RTI.

Modifying Distributed Region Extents

The three regions used for implicit DDM are updated based on the changes to the locations of the federate's objects. Each object has an update region and a weapon region. A single subscription region is maintained for the federate, but the region has an extent for each registered object. The update and weapon region as well as the subscription extent are checked whenever the corresponding object's location changes. The implicit DDM uses ranges and margins that can reduce the actual changes significantly, which is important with distributed regions. (For details, see "12.1.4 Configuring Implicit DDM" on page 171.)

When using the distributed region approach the update region reflects the furthest distance the object instance could move during a single update interval (the update range). The region has a single extent which surrounds the object. The range of the extent can be configured based on the class used to register the object instance. It assumes that the object can change direction instantaneously, including 180° reversals. In other words the object's current orientation and velocity (momentum) are ignored in calculating the object's range. This simplifies the determination of region boundaries when processing location updates.

The object's weapon region reflects the range at which an object can interact with another (that is, sending a weapon fire or detonation interaction). Its range is an extension of the update range.

The subscription extent reflects the furthest range at which an object can detect another object (the sensor range). The sensor range is also an extension of the object's update range.

The Implicit DDM plug-in allows all range values to be configured on a class-by-class basis, so update region, interaction region, and subscription region sizes can be tailored to the type of object (vehicle, aircraft, surface vessel, and so on).

When any `BaseEntity`-class object performs an attribute value update, the plug-in decodes the `WorldLocation` attribute if it is included in the update and compares it against the three regions' extents, updating any of them if necessary. The region modifications are initiated before the attribute value updates are processed further. Aside from initiating the region modifications, the plug-in does no further special processing on the attribute value updates. The RTI's DDM processing checks for overlaps with other known objects' regions and establishes the communications channels.

To minimize the number of region modifications that must be performed, the plug-in can expand the boundaries of the regions by a factor that you can configure. This is particularly beneficial when using distributed regions in which each region modification results in one or more messages being sent over the network to be processed by all

federates. This is less important when fixed grid DDM is used, because region modifications do not prompt additional network traffic or require processing at the remote federates.

When a federate sends an interaction using `sendInteraction()`, the plug-in decodes the `FiringObjectIdentifier` parameter and looks in a map of object instance names to determine if the interaction was sent by a `BaseEntity` object which has an interaction region associated with it. If an interaction region can be associated with the interaction, the plug-in simply substitutes the `sendInteractionWithRegion()` call to the RTI for further processing.

i If you use reliable internal messaging, it is strongly recommended that you enable the **RTI_dualTransmitFirstInteractionRegions** RID file parameter to compensate for a potential race condition between the region modification messages and the interaction message.

12.4.5 Modifying Fixed Grid Region Extents

When fixed grid DDM is enabled, the update and interaction regions become point regions rather than volumes. The update range and weapon range do not affect the region scope. Instead, when the federate updates an object's location attribute the update region's extent is set to the single point specified by the location. No change is made to the interaction region. This single point is enough for the underlying fixed grid implementation to establish a communication channel.

The interaction region is also handled differently in that the region is based on the location of the interaction, rather than the location of the object that initiated the interaction. The plug-in decodes the `FiringLocation` parameter (for `WeaponFire` interactions) or `DetonationLocation` parameter (for `MunitionDetonation` interactions), and the `FiringObjectIdentifier` parameter from the interaction message data. It looks in the map of the registered object instance to retrieve the interaction region for the object that sent the interaction. It then modifies that interaction region to reflect the interaction location. Again, this single point is sufficient for the fixed grid implementation to establish a communication channel.

The subscription region is also modified when the federate updates the location of one of its objects. The object's subscription region extent represents a volume determined by the corresponding object's location extended in all dimensions by the sensor range. The subscription region volume results in the local RTI listening to the communication channels where the sensor range overlaps the update or weapon range.

Fixed Grid DDM does not exchange region modification messages, so additional bandwidth savings can be realized.

i The bandwidth savings will be offset somewhat by the fact that the smart forwarding feature is incompatible with fixed grid DDM, so all updates and interactions will transit the network even when there are no subscribers joined to the same multicast group.



13. Using FOM Modules

Use of FOM modules is functionality from the HLA Evolved specification. It has been implemented in the MAK RTI so that it can be used in RTI 1.3 and 1516 federations.

13.1. Using FOM Modules	188
13.2. Merging FOM Modules into the Current FOM	188
13.2.1 Using FOM Modules in Lightweight Mode	189
13.3. Creating and Joining with FOM Modules	190
13.4. Validating FOM Modules	190
13.5. Configuring Use of Modular FOMs	191
13.5.1 Configuring a MOM Module	192
13.5.2 Configuring Create FOM Modules	192
13.5.3 Configuring Join FOM Modules	192
13.5.4 Configuring Use of a Local FOM Module File	193

13.1. Using FOM Modules

The MAK RTI allows a federation to be formed and modified using FOM modules. The use of FOM modules allows federates with different FOMs to interoperate without having to merge the FOM files. The different FOMs must use the same reference FOM, and have non-conflicting extensions. Using Modular FOMs supports more agile federations, because only FOM modules pertinent to a particular federate or subset of federates need to be introduced into the federation. Federates can specify one or more FOM modules when creating a federation, and more significantly, a federate can specify one or more FOM modules when it joins. Examples of FOM modules are:

- A normal, fully formed FOM. A federate can create a federation using a FOM just as it always has in the past. It can also submit a FOM when joining, to ensure that the classes it uses are included in the federation.
- A fully formed FOM that is an extension of an existing FOM. It is common to add new classes to a reference FOM to support new simulation concepts or extensions to existing concepts. The use of FOM modules allows these FOM extensions to be introduced by various federates. The extensions do not have to be integrated into a single superset FOM. However, the extensions must not conflict with each other.
- A partial FOM containing new or derivative concepts. New or derivative concepts are supplied in a self-contained FOM module that is not a complete FOM. The new or derivative concepts might depend on concepts found in an existing FOM (especially in the case of derivative concepts).

 Modular FOMs is not part of either the 1.3 or the 1516 API, so the MAK RTI configures its use through configuration parameters in the RTI settings file. HLA Evolved and HLA 4 federates can specify FOM modules during create and join using the HLA API.

13.2. Merging FOM Modules into the Current FOM

To help understand how FOM modules work together in an HLA 1.3 or 1516 federation we use the concept of the "current FOM". The current FOM is the result of merging all the FOM modules that have been submitted by federates. The RTI starts the current FOM with the basic MOM classes. Then it merges in the FOM modules specified by the creating federate and joining federates. (In HLA Evolved federations, modular FOMs work as defined in the specification.)

The MAK RTI supports three types of FOM modules – the MOM module, the create FOM module, and the join FOM module. A FOM module must satisfy specific merging rules when submitted to the RTI. A valid FOM module can extend the current FOM only by adding subclasses. It cannot extend an existing class by adding attributes or parameters, or changing the datatype or any other FOM characteristic of an existing class or its attributes or parameters. All FOM Modules to be merged must have identical time types and switches.

The MOM module is a special case that relaxes the restrictions on adding class attributes and parameters to existing MOM classes. It is explicitly stated in the HLA standard that the MOM classes can be extended, including adding attributes and parameters. Because the

initial current FOM consists of the base MOM classes, the rules for adding class attributes and parameters to existing base MOM classes are allowed, but only through the MOM module. In effect, a MOM module replaces the base MOM classes and it becomes the initial current FOM. It is validated to ensure that it contains the necessary classes for the RTI to operate.

The federate that creates the federation can specify one or more create FOM modules in its RID file. It can also specify one MOM module.

 In the HLA Evolved specification, the term MOM module, used in early drafts and therefore in the initial implementation of modular FOMs in the MAK RTI, was changed to MOM and Initialization Module, or MIM. If FOM module merging is disabled, a MIM cannot be used. If FOM module merging is enabled, a MIM is always used. The federate can specify the MIM it would like to use in the createFederationExecution call. If it does not specify the MIM, the standard MIM is used. The standard MIM is `./bin/HLAstandardMIM.xml` for HLA Evolved and `./bin/HLAstandardMIM-2025.xml` for HLA 4. When running a mixed HLA Evolved and HLA 4 federation, the standard MIM merged into the "current FOM" depends on which protocol creates the federation. See "1.4.1 RTI Interoperability" on page 23 for more information.

 Merging FOM modules is non-commutative and non-associative. Therefore, the order in which FOM modules are specified may affect whether or not merging a FOM module with the current FOM is a valid operation. The same is true for join modules. This is true both for the order that modules are specified in a particular RID file and the order in which federates join the federation.

13.2.1 Using FOM Modules in Lightweight Mode

The MAK RTI supports FOM modules in lightweight mode. The general conditions of lightweight mode are the same. All federates must use the same FOMs and they must all match. Some federates may require different FOM modules or have their FOM module requirements hard coded. In the absence of an rtiexec, there is no way to programmatically determine the FOM module requirements of all federates. Therefore to ensure that all federates use the same FOMs, to use FOM modules in lightweight mode, you must specify the module requirements using RTI settings. In the RTI initialization data (RTI settings) file for each federate, use the following parameters to specify the FOM modules required by the federation.

```
;; (RTI-addCreateFomModule "creationFomModule.xml")
;; (RTI-addJoinFomModule "JoinFomModule.xml")
;; (setqb RTI_momModuleExtensionFileName "momExtension.xml")
```

For example, suppose federate A needs FOM module A and federate B needs FOM module B. They both need to specify both FOM modules in their RID files:

```
(RTI-addJoinFomModule "A.xml")
(RTI-addJoinFomModule "B.xml")
```

13.3. Creating and Joining with FOM Modules

When the FOM module feature is enabled, the MAK RTI enhances the behavior of the `createFederationExecution` call by merging FOM modules in addition to the FOM specified in the create call.

For HLA 1.3 and 1516, the FOMs are merged in the following order:

1. The MOM module (if specified in the RID file).
2. The FED file specified in the create call.
3. The create FOM modules, in the order specified.

For HLA Evolved and HLA 4, FOMs are merged in the following order:

1. The MIM.
2. The FOM modules:
 - When FOM Sorting is enabled, the RTI sorts the FOM modules before merging.
 - When FOM Sorting is disabled, the RTI merges the FOM modules in the order specified.

To enable or disable FOM Sorting, change the value of the **RTI_fomModuleSorting** in the RTI settings. (See Modular FOMs in Table 13 on page 218.) By default, FOM Sorting is enabled.

The join FOM modules are introduced for merging in a similar fashion. When a federate joins, there is already a current FOM consisting of the base MOM (or MOM module) and at least one FOM module (the FOM submitted in the create call). The join FOM modules are merged into the current FOM in the order specified in the RID file.

If you use FOM modules, the RTI must distribute the FED file (specified by the **RTI_distributeFedFile** parameter). This ensures that every federate has the complete set of FOM data being used by the federation. The use of FOM modules also requires the use of the `rtiexec` and that internal messages be sent reliably (**RTI_useRtiExec** and **RTI_internalMsgReliableWhenUsingRtiexec**).

If a federate tries to create a federation that already exists, any FOM modules it specifies have no effect on the current FOM. Additionally, the create and join operations, including the merging of the FOM modules, are an atomic action. A failure to merge a single FOM module results in the failure of the entire call and the current FOM does not change.

13.4. Validating FOM Modules

A FOM module merge requires that the time type and switches tables be identical. A valid merge also requires that a class definition must be complete. A valid module cannot extend a class with new attributes or parameters that have already been defined in another module; it can only extend the FOM by adding new classes.

If a module contains definitions of classes that already exist in the FOM, they must be identical to the existing class definitions or they must be scaffolding definitions. A scaffolding class does not replace existing class definitions nor does it have to match them.

However, if a FOM with scaffolding classes is introduced out of order, its empty class definition takes precedence and subsequent complete class definitions will be flagged as an invalid merge.

 A scaffolding definition is one that describes the class hierarchy and inheritance information of a class without specifying any attributes or parameters of the class. Since an object class definition must include its entire parentage, but may not be specifying the object classes of its parentage, often a FOM module will include so-called scaffolding class definitions, which do not include any object or interaction class information beyond the hierarchy.

To restate the basic merging rule, once an object or interaction class is added to the current FOM, its attributes and parameters cannot be changed. Merging a non-scaffolding object or interaction class definition requires that the object or interaction class be absent in the current FOM or, if present, match the existing class attributes and parameters exactly.

HLA Evolved and HLA 4 have different FOM module merging requirements. Above, we detail the requirements of the HLA Evolved specification. HLA 4 has removed most of the restrictions of FOM extension set in place by HLA Evolved. HLA 4 allows extending an existing class with new attributes or parameters that have already been defined in another module. This applies to datatype definitions as well. For more details about the HLA 4 merging rules, see the IEEE 1516-2025 OMT specification.

13.5. Configuring Use of Modular FOMs

The use of FOM modules is configured in the RTI initialization data (RTI settings) file with the **RTI_fomModuleMerging** parameter.

For HLA 1.3 and 1516, the presence of any of the FOM module functions in the RID file enables the FOM module functionality. For HLA Evolved, **RTI_fomModuleMerging** must be enabled to use FOM modules. The use of FOM modules also requires enabling the `rtiexec`, internal message reliable transport, and FED file distribution.

It is not necessary for each federate to have the same entries in its RTI settings file. However, the use of any FOM module must adhere to the modular FOM merging rules described in "13.2. Merging FOM Modules into the Current FOM" on page 188.

- 
- The previous paragraphs in this section apply to all three HLA specifications. However everything else in this section, including the following paragraph, applies only to HLA 1.3 and 1516.
 - Not all modular FOM configuration parameters affect a federation. For example, **RTI-addCreateFomModule** and **RTI_momModuleExtensionFileName** only affect the federation if the federate creates the federation. A MOM module extension or create FOM module of a federate that joins an already created federation have no effect on the federation.

13.5.1 Configuring a MOM Module

To specify a MOM module file, use the **RTI_momModuleExtensionFileName** parameter in the RTI settings file. To be executed, the federate must create the federation execution. The syntax is:

```
(setqb RTI_momModuleExtensionFileName "momModuleName")
```

where *momModuleName* is a quoted string that specifies the MOM module file name. The file name can be an absolute path, a path relative to the working directory, or just the filename if it is in the working directory.

The specified MOM module can contain extensions to the existing MOM classes. The module specified in this parameter can be considered the first FOM module specified, because it will be merged into the current FOM before any other modules specified in the RTI settings file. This parameter is optional and the default base MOM classes are used if omitted.

13.5.2 Configuring Create FOM Modules

Create FOM modules are FOM modules specified by the federate that creates the federation. To specify a create FOM module, use the **RTI-addCreateFomModule** function in the RID (RTI initialization data, or RTI settings) file. The syntax is:

```
(RTI-addCreateFomModule "fomModuleName")
```

where *fomModuleName* is a quoted string that specifies a FOM module file name. The file name can be an absolute path, a path relative to the working directory, or just the filename if it is in the working directory.

Each function specifies a single create FOM module. You can include multiple entries in the RTI settings file. The order that they appear in the file is the order they are merged into the current FOM. The RTI will try to merge the contents of each FOM module into the current FOM during the create federation execution service call. If successful, the FOM module becomes part of the current FOM supplied to federates when they join. If unsuccessful, the create call fails.

13.5.3 Configuring Join FOM Modules

Join FOM modules are FOM modules specified by a federate that is joining a federation. To specify a join FOM module, use the **RTI-addJoinFomModule** function in the RTI settings file. The syntax is:

```
(RTI-addJoinFomModule "fomModuleName")
```

where *fomModuleName* is a quoted string that specifies a FOM module file name. The file name can be an absolute path, a path relative to the working directory, or just the filename if it is in the working directory.

Each function specifies a single join FOM module. You can include multiple entries in the RTI settings file. The order they appear in the file determines the order that they are merged into the current FOM. The RTI will try to merge the contents of each FOM module

into the current FOM during the join federation execution service call. If successful, the FOM module becomes part of the current FOM supplied to other federates when they join. If unsuccessful, the join call fails.

13.5.4 Configuring Use of a Local FOM Module File

When a federate submits a FOM module in a create or join call, the file is transmitted to the rtiexec. The rtiexec sends the FOM module back to the federate when it joins. (The FOM module is also sent to federates that have already joined and to subsequent joiners.) A federate can use its local copy of a FOM module instead of having it sent back from the rtiexec. The **RTI_preferLocalFomModule** parameter controls whether a LRC uses its local copy or requests a new copy from the rtiexec for any FOM module files that it has submitted. The syntax is:

```
(setqb RTI_preferLocalFomModule enableOption)
```

where *enableOption* is 1 to enable (use local copy) and 0 to disable (rtiexec retransmits copy).



14. Update Rate Reduction

Update rate reduction is functionality from the HLA Evolved specification. It has been implemented in the MAK RTI so that it can also be used with RTI 1.3 or RTI 1516 federates.

14.1. Update Rate Reduction	196
14.2. How Update Reduction Is Implemented	197
14.3. Configuring Update Rate Reduction	198
14.3.1 Specifying the Update Rate in 1.3 and 1516	199
14.3.2 Specifying the Rate for Object Class Subscriptions in 1.3 and 1516 ...	200
14.3.3 Specifying the Receive Filtering Tolerance	200
14.3.4 Example Update Rate Reduction Configuration (1.3 and 1516 Only) .	200

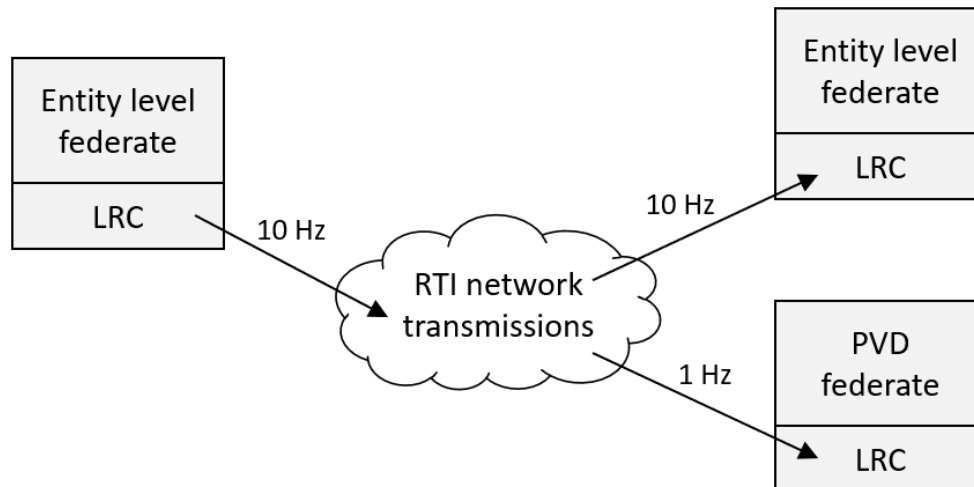
14.1. Update Rate Reduction

Update rate reduction is a mechanism that allows federates to specify the rate at which best effort attribute updates can be delivered to the federate. This feature augments the declaration management (DM) and data distribution management (DDM) services that provide mechanisms for reducing or filtering the data that a federate receives based on class and data content. Update rate reduction does not affect interactions or reliable attribute updates.

The DM and DDM services provide mechanisms for reducing or filtering the data that a federate receives. The DM services provide filtering on the type of data; the DDM services provide filtering on the content or value of the data (that is, whether the value of the data is within a given range). It may be that even with these powerful filtering mechanisms, a federate might still not be able to keep up with the data that meet its filter criteria. In these cases, the federate would be better off receiving some sampling of the data than being overloaded, falling behind, and overflowing its buffers.

An example of this requirement could be a low fidelity plan view display federate that is viewing an entire simulation. It is not interested in the detailed maneuvers and interactions of the simulation, but in the general macro view of the entity placement and long term movements. While this federate may require a sample of all the data (both type and value), it does not require the data at the resolution that is necessary for entity level simulations. Figure 60 illustrates the update rate reduction for this example.

Figure 60. Update rate reduction



By using update rate reduction, the federate can specify a hard limit on the rate at which the RTI delivers attribute updates from each object instance. The RTI is responsible for delivering attribute updates for each object to the federate up to the specified rate. There is no guarantee that the RTI will send attribute updates at a given rate, just that the rate will not be exceeded. The criterion used for delivering updates is based on the specified rate and message inter-arrival times. Given a maximum update rate of x Hz and a message that has been delivered at wall-clock time t , no message shall be delivered before $t + (1 / x)$.

The update rate applies to each object attribute instance independently. That means that in practice, the rate that attributes can be reflected (where a reflection contains the attributes from one object) is the number of discovered objects times the update rate.



- The maximum rate would actually be:

$$\text{number_of_attributes} * \text{number_of_objects} * \text{rate}$$
 However, this maximum is a degenerate case in that it would require the sending federates to update all the object attributes (in practice just a few) each independently of the others (in practice in a single update).
- If no update rate reduction is specified for a federate, the federate will receive attribute updates at whatever rate they are sent.

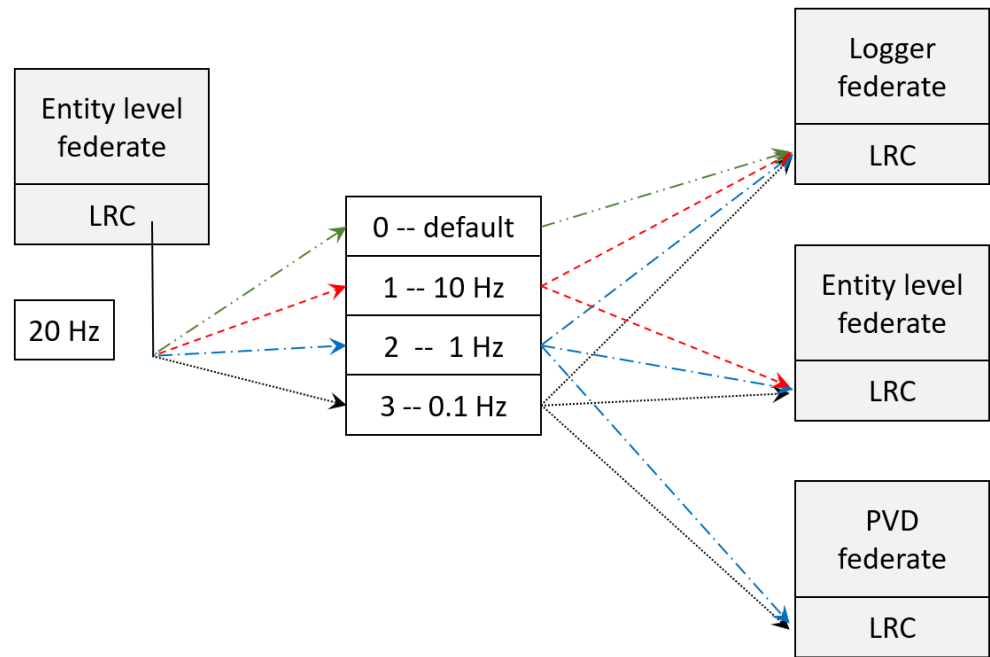
14.2. How Update Reduction Is Implemented

The MAK RTI implements update reduction using multicast and receive filtering. The implementation relies on the DM multicast filtering feature that allows object classes to be assigned multicast addresses. (For details, please see "9.12. Filtering by Multicast Groups Using Declaration Management" on page 132.) A set of update rates is then defined in the RID file ("14.3. Configuring Update Rate Reduction" on the next page) and each rate is assigned an offset. The combination of an object class's multicast address and update rate offset determine an update rate channel.

When sending an attribute update message, the RTI inspects the set of attributes and determines the interval since each attribute was sent. It selects the greatest interval, and sends the update to the appropriate update rate channel.

On the subscriber side, when a federate subscribes to object class attributes, the RTI inspects the update rate subscription specified for the subscribed class. The RTI then joins the update rate channels for the specified rate and all slower rates. The result is that an update sent at the specified rate or slower will be received by the federate and the sum of the updates on all the update rate channels will equal the subscribed update rate, as illustrated in Figure 61. This multicast filtering provides an efficient means for the LRC to receive and process only those update messages that will be delivered to the federate. However, to compensate for timing issues, the LRC also performs receive side filtering to further guarantee the specified rate.

Figure 61. Update rate routing



The arrows in Figure 61 represent the information as shown in the table.

UA at t, t+10., ...	Black arrow with dotted line→
UA at t+1.0, 2.0, ...	Blue arrow with a dash and a dot - . - . - . →
UA at t+0.2, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.01, 1.02, ...	Red arrow with dashes - - - - - →
UA at t+0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 1.05, 1.15, ...	Green arrow with a dash and two dots - . . - . . →

14.3. Configuring Update Rate Reduction

To use update rate reduction, you must specify a set of update rates and associate them with object class attribute subscriptions. You must also configure use of DM multicast filtering. The update rates are specified as named values given in hertz (updates per second). These named values are then associated with object classes with a nesting operator similar to that used in the configuration of DM multicast filtering (that is, inclusive indicates the specified class and all subclasses and exclusive indicates just the specified class). The DM multicast addresses must be assigned to leave room for the update rate offsets, including a base address of no rate.

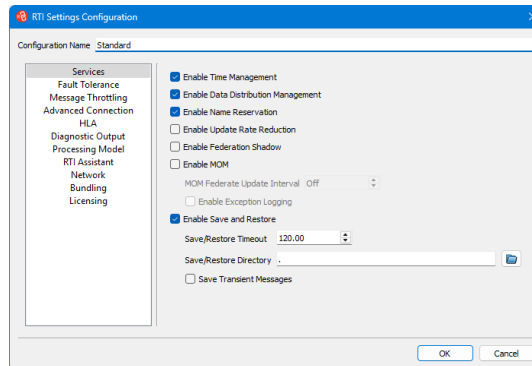
For example, if three update rates are specified, then each DM multicast assignment must have at least 4 addresses separating it from any other (for example, 229.7.7.10, 229.7.7.14, 229.7.7.18, and so on). You can specify a receive tolerance to allow fine tuning of receive filtering versus multicast filtering.

The update rate reduction is configured in the RTI settings.

► **To enable update rate reduction:**

1. If the RTI Assistant is not running, on the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Utilities**. In the Utilities folder, double-click the RTI Assistant shortcut.
2. In the system tray, right-click the RTI Assistant icon (🔌). A context menu displays.
3. Choose **Preferences**. The RTI Preferences dialog box opens.
4. Select the RTI Settings Page (Figure 1 on page 44). It lists RTI settings configurations provided by MAK and any configurations that you may have created.
5. Select the settings configuration that you want to edit.
6. Click the Edit button (✎). The RTI Settings Configuration dialog box opens. It has a set of pages with configuration settings.
7. Select the Services page (Figure 62).

Figure 62. Services page



8. Select the Enable Update Rate Reduction check box.
9. Click OK.

For detailed explanation of DM multicast configuration, see "9.12. Filtering by Multicast Groups Using Declaration Management" on page 132.

14.3.1 Specifying the Update Rate in 1.3 and 1516

- Specify update rates with the function **RTI-addUpdateRate**. Include an entry for each rate:

```
(RTI-addUpdateRate "name" rate)
```

where *name* is a quoted unique string and *rate* is a floating point value specifying the rate in hertz.

i In HLA Evolved, the update rates are specified in the FDD file.

14.3.2 Specifying the Rate for Object Class Subscriptions in 1.3 and 1516

You can specify an update rate for object class subscription with the function **RTI-addUpdateRateSubscription** in the RTI settings file. Add an entry for each class that should have update rate reduction applied to its subscription. The nesting qualifier (*inclusive* or *exclusive*) is a shortcut that allows the rate to be applied to all subclasses of the specified class. The entries are applied in the order they appear in the RTI settings file, and the nesting operators override previous operations. For example, a top level class could specify a rate as *inclusive*, meaning it applies to all subclasses. Selected subclasses could override that rate with *exclusive* entries (that apply only to themselves), or *inclusive* entries (that apply to themselves and their subclasses).

The syntax for **RTI-addUpdateRateSubscription** is:

```
(RTI-addUpdateRateSubscription "class" "rateName" "nesting")
```

where:

- *class* is a quoted string specifying an object class name, which must be in the joined FOM.
- *rateName* is a quoted string specifying the name of an update rate, which must be specified in an **RTI-addUpdateRate** entry previously in the RTI settings file.
- *nesting* is a quoted string specifying either *inclusive* or *exclusive*, where *inclusive* indicates the rate applies to the specified class and all subclasses and *exclusive* indicates the rate applies to just the specified class.

 In HLA Evolved, the subscription rate is specified using the API when a federate subscribes to object class attributes.

14.3.3 Specifying the Receive Filtering Tolerance

Specify the receive-filtering tolerance with the **RTI_receiveTolerance** parameter. This parameter is a percentage reduction to the inter-arrival time between attribute updates. It reduces the inter-arrival time for a given rate so that an update that appears to arrive early at the receiver is accepted. Given a rate of 10 hertz, a tolerance of 0.1 would allow the LRC to accept an update arriving 0.09 seconds after a previous update. This tolerance value allows for differences in timing measurements between sending and receiving federates as well as allowing for latency fluctuations (that is, previous update was late, but the next is received faster). The syntax for **RTI_receiveTolerance** is:

```
(setqb RTI_receiveTolerance reductionPercentage)
```

where *reductionPercentage* is a floating point value specifying the percentage to reduce the inter-arrival interval used for receive filtering.

14.3.4 Example Update Rate Reduction Configuration (1.3 and 1516 Only)

Here is an example configuration of update rate reduction using all the necessary parameters:

```
(setqb RTI_enableUpdateRateReduction 1)
(RTI-addDMObjectMulticastAddr "ObjectRoot.BaseEntity" "229.7.8.1"
 "inclusive" "0.0.0.0")
```

```
(RTI-addDMObjectMulticastAddr "ObjectRoot.BaseEntity.GroundVehicle"
  "229.7.8.5" "exclusive" "0.0.0.0")
(RTI-addDMObjectMulticastAddr "ObjectRoot.BaseEntity.Aircraft"
  "229.7.8.9" "exclusive" "0.0.0.0")

(RTI-addUpdateRate "high" 10.0) ;; up to 10 updates per second
(RTI-addUpdateRate "medium" 1.0) ;; up to 1 update per second
(RTI-addUpdateRate "low" 0.1) ;; up to 1 update every 10 seconds

(setqb RTI_receiveTolerance .01)

(RTI-addUpdateRateSubscription "ObjectRoot.BaseEntity" "low"
  "inclusive")
(RTI-addUpdateRateSubscription "ObjectRoot.BaseEntity.GroundVehicle"
  "medium" "exclusive")
(RTI-addUpdateRateSubscription "ObjectRoot.BaseEntity.Aircraft"
  "medium" "exclusive")
```




A. RTI Settings Reference

This appendix describes the settings in the *rid.mtl* file.

A.1. MÄK Technologies Lisp (MTL) Files	204
A.1.1 Using Environment Variables in an MTL File	205
A.1.2 Using Conditional Statements	205
A.1.3 Converting Between Strings and Numbers	205
A.2. RTI Settings	205
A.3. Federation-Wide Parameters	206
A.4. LRC-Specific Parameters	217
A.5. Configuring tick(min,max)	233

A.1. MÄK Technologies Lisp (MTL) Files

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

 The MÄK Technologies Lisp (MTL) files used by MAK ONE products to specify configuration settings are parsed at startup by the RTI. When a configuration file has certain errors, such as unmatched parentheses, the parser will not necessarily return an error code. This can cause the RTI to reach a partially initialized state or cause the RTI to initialize itself with default RID values, causing unexpected behavior. This applies to lisp expressions in the RID file as well as the expressions passed to the 1516 createRTIambassador function.

Most parameter entries take the format:

```
(setq parameter_name value)
```

For example:

```
(setq Enablewireframe 1)
```

or:

```
(setqb parameter_name value)
```

The `setqb` syntax indicates a bound variable.

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

In most cases, the parameter names are known to the MAK application and the MTL commands simply assign values to them. However, you can use the `setq` command to create arbitrary symbolic names or aliases that are then used in later commands. For example, in the following two commands the symbolic name `plugin-directory-root` is assigned to a directory path. Then the symbolic name is used in the `if` command. The name `plugin-directory-root` has no intrinsic meaning to the application. It is given meaning by the `setq` command, which allows it to be used later in the `if` command.

```
(setq plugin-directory-root "../plugins/B-HAVE/")
(if (equal Transport "DIS") (setq plugin-directory
    (string-append plugin-directory-root "binDIS/")))
```

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

```
(load "C:\MAK\MAKproduct\config\config.mtl")
```

You could also create a variable to hold the path and load files:

```
(setq rid-path "C:\MAK\MAKproduct\config\")
(load (append-string rid-path "base-rid.mtl"))
```

A.1.1 Using Environment Variables in an MTL File

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

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

For example:

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

A.1.2 Using Conditional Statements

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

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

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

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

A.1.3 Converting Between Strings and Numbers

MTL has the following functions for converting strings to numbers and numbers to strings:

- `string->number` converts a string to a number.
- `number->string` converts a number to a string.

Examples:

```
(setqb RTI_udpPort (string->number "4000"))
```

or

```
(setqb someLispStringVariable (number->string 6382))
```

You can chain these with other commands as in this example:

```
(setqb RTI_udpPort (string->number (getenv (quote
RTIEXEC_PORT))))
```

A.2. RTI Settings

The MAK RTI RTI settings (RID) file uses two formats for parameters. Most parameters are in the format:

```
(setqb RTI_parameterName value)
```

This format behaves like an assignment in which the value is assigned to *RTI_parameterName*. Repeated occurrences of an assignment parameter result in its taking the value from the last occurrence. Assignment parameters begin with **RTI_**.

Some parameters behave more like a function, in which the function is applied to the value. Repeated occurrences of these function parameters usually result in the values from each occurrence being accumulated. The function parameters begin with **RTI-** and they omit the `setqb` keyword as in this example:

(RTI-parameterName value)

A default *rid.mtl* file is in the MAK RTI root directory.

A.3. Federation-Wide Parameters

Table 12 lists and briefly describes parameters in *rid.mtl* that must be the same for every federate in a federation. The table is organized to match the *rid.mtl* organization. If applicable, the description includes a cross reference to further information in the manual.

Many of these parameters can be set in the RTI Settings page of the RTI Preferences dialog box, which is the preferred method for editing them. For details, see "4.3. Editing RTI Settings" on page 43.

Table 12. Exercise-wide RID file parameters

Parameter	Description
RTI Compliance	
RTI_forceFullCompliance	Specifies all relevant RID file parameters to support full compliance with the HLA Interface Specification. 0 – Do not force full compliance. 1 – Force full compliance. Default: 0. For more information, see "11.1. Enabling and Disabling HLA Services" on page 160. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.
Communications Configuration	
RTI_useRtiExec	Determines whether the application should try to connect to an rtiexec application when it creates, destroys, or joins a federation execution. 0 – Do not use the rtiexec. 1 – Use the rtiexec. Default: 0. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_tcpPort	Specifies the port number to be used for TCP messages. Default: 4000. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.
RTI_udpPort	Specifies the port number to be used for UDP messages. Default: 4000. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.
RTI_destAddrString	Specifies the IP address to which best effort federation execution traffic is sent. This will typically be a broadcast address or multicast address. Default: "229.7.7.7" (a multicast address). A value of "0.0.0.0" is a special value that indicates that the default broadcast address should be used. If any of your federates are running on platforms on which multicast is not supported, you must use a broadcast address rather than a multicast address. For more information, see "10.1.1 Configuring Best Effort (UDP) on a LAN" on page 136. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.
RTI_tcpForwarderAddr	Specifies the IP address of the machine that the rtiexec application is running on. This value is required if you are using reliable transport. Default: "127.0.0.1" For more information, see "10.1. Configuring UDP and Centralized TCP Forwarding on a LAN" on page 136 and "10.2. Configuring UDP and Centralized TCP on a WAN" on page 137. This parameter is ignored unless RTI_configureConnectionWithRid is set to 1.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_internalMsgReliableWhenUsingRtiexec	Specifies whether or not you want to send the RTI's internal bookkeeping messages using reliable transport. This parameter is ignored if RTI_useRtiExec is set to 0. 0 – Use best effort for all internal messages. 1 – Use reliable for all internal messages. Default: 1. For more information, see "9.1. Choosing Best Effort or Reliable Transport" on page 122. This parameter is set to 1 unless RTI_configureConnectionWithRid is set to 1.
RTI_fomDataTransportTypeControl	This parameter sets the FOM Data Transport Type Controller: 0 – FOM data is sent either best effort or reliable based on FOM settings. This is the default. It is forced when forceFullCompliance is enabled. 1 – All FOM data is sent best effort. It is the forced setting when using lightweight configurations. 2 – All FOM data is sent reliable. This is permitted only when RTI_configureConnectionWithRid is 1.
Optional Services	
RTI_momServiceAvailable	Enables (1) or disables (0) management object model (MOM) services. Default: 0. For more information, see "11.1.1 Configuring MOM Services" on page 160
RTI_timeMgmt	Enables (1) or disables (0) time management services. Default: 0. For more information, see "11.1.2 Configuring Time Management Services" on page 162.
RTI_nextEventRequestAndRetractServiceEnabled	Enables (1) or disables (0) next message request and retract services. Only meaningful if RTI_timeMgmt is not enabled. Default: 1.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_extend13and1516interop	Extends interoperability between 1.3 and the 1516 versions (for example, handling DDM dimensions). For more information, see "1.4.1 RTI Interoperability" on page 23.
RTI_dataDistMgmt	Enables (1) or disables (0) data distribution management (DDM) services. Default: 0. For more information, see "12.1. Configuring DDM Services" on page 168.
RTI_ddmFixedGrid	Enables (1) or disables (0) fixed grid DDM versus distributed region DDM when DDM is enabled. Default: 0. For more information, see "12.1.3 Configuring a Fixed Grid DDM" on page 169.
RTI_ddmFixedGridFilename	Specifies the fixed grid configuration file name. For more information, see "Fixed Grid Search Order" on page 171.
Service Implementation	
RTI_responseInterval	Specifies the amount of time (in seconds) that the local RTI component waits to receive a response from the rtiexec when processing the createFederation, destroyFederation, or joinFederation service calls. Default: 3 seconds. For more information, see "6.8.3 Specifying the Response Interval" on page 95.
RTI_useRandomNumberForFedHandle	Instructs the RTI to create federate IDs based on random numbers rather than the federate's process ID, when not using the rtiexec. 0 – Use process ID. 1 – Use pseudo random number. Default: 0.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_enableHlaObjectNamePrefix	Specifies that the RTI, and only the RTI, will create object names that begin with "hla". 0 – Do not force object names to start with "hla". 1 – Force object names to start with "hla". Default: 0.
RTI_sendDiscoveredClass	Enables (1) or disables (0) transmission of discovered class advisory messages. Default: 0. To meet the interface specification requirements for the turn updates on or off advisories, the discovery class of remote objects must be relayed to federates that own attributes of those objects. The transmission of the discover messages can be disabled, reducing the overhead with only a slight degradation of the advisory functionality (the actual class is used instead of the discovered class).
RTI_enableFomBackwardsCompatibility	Enables (1) or disables (0) use of the handle numbering scheme used in MAK RTI 2.3.3 and earlier. It supports federates that recorded old handles in a file. Default: 0.
RTI_reuseReleasedObjectHandles	Allows the RTI to reuse object instance handles if the original instance has been deleted. Use of this feature breaks the guarantee that handles are unique over the lifetime of a federation. Handles will still be unique at any given instant.
RTI_distributeFedFile	Enables (1) or disables (0) distribution of the FED or FDD file. Default: 0. The federation must also enable RTI_useRtiExec and RTI_internalMsgReliableWhenUsingRtiexec. For more information, see "6.2.2 Distributing the FED File" on page 84.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_fullFedFileDistribution	Federates and the federation execution will only request FOMs not locally available if set to 0. If set to 1, the federation execution and federates will distribute all FOM files to all federates and federations even if they already have it.
RTI_enableNameReservation	Enables (1) or disables (0) the name reservation service in HLA 1516 or HLA Evolved. Default: 1.
RTI_strictNameReservation	Enables (1) or disables (0) use of strict 1516 name reservation. If enabled, it ensures that no more than one object can reserve a name for the lifetime of the federation. Default: 0.
RTI_strictHLA1516eAttributeOwnershipUnavailable Callback	Forces a strict attribute ownership unavailable callback. In HLA Evolved, enabling this parameter may cause attributeOwnershipUnavailable to not occur in federates with pending acquisitions in response to the cancelNegotiatedAttributeOwnershipDivestiture and confirmDivestiture service calls when the federate does not receive the requested attributes. Default: 0.
RTI_automaticNameReserveEnabled	Allows HLA version 1.3 to internally, and automatically, reserve the name of any object it registers. This slows down registering new objects because it requires a call to the RTIexec. However, it guarantees no duplicate names in objects. If objects are registered within close proximity by different federates they may be assigned the same name.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
Fault Tolerance	
RTI_deleteOrphans	When the federation shadow is disabled, specifies whether the objects for which a terminated federate owns the <code>privilegeToDelete</code> will be automatically deleted. 0 – disabled (all owned attributes are orphaned). 1 – enabled. Default: 1. For more information, see "Handling Orphaned Attributes" on page 92.
RTI_enableFederateHeartbeat	Enables (1) or disables (0) the requirement that federates send a heartbeat.
RTI_federateHeartbeatInterval	Specifies the frequency, in seconds, with which each federate LRC sends a heartbeat message. Default: 10. Valid only if RTI_enableFederateHeartbeat is set to 1. For more information, see "6.8.1 Responding to Abnormal Termination" on page 91.
RTI_federateTimeoutInterval	Specifies the time period in which the federation must receive a heartbeat or it will forcibly remove the federate. Default: 25. Valid only if RTI_enableFederateHeartbeat is set to 1. For more information, see "6.8.1 Responding to Abnormal Termination" on page 91.
RTI_federationShadowEnabled	Specifies whether a federation shadow process is started for each federation to handle ownership of orphan attributes when a federate terminates abnormally. Default: 0. For more information, see "Handling Orphaned Attributes" on page 92.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_reconnectEnabled	Specifies that the LRC try to reconnect to the RTI Forwarder RTI_federateReconnectPause seconds after a TCP connection is lost. Default: 0. For more information, see "6.8.2 Reconnecting to the RTI Forwarder" on page 94.
RTI_federateReconnectPause	If RTI_reconnectEnabled is enabled, specifies the amount of time, in seconds, to wait before trying to reconnect to the RTI Forwarder. Default: 5. For more information, see "6.8.2 Reconnecting to the RTI Forwarder" on page 94.
RTI_rtiExecReconnectPause	If RTI_reconnectEnabled is enabled, specifies the amount of time, in seconds, to wait before trying to reconnect to the RTI Forwarder. Default: 4. For more information, see "6.8.2 Reconnecting to the RTI Forwarder" on page 94.
Advanced Network Configuration	
RTI_use32BitsForValueSize	Enables (1) or disables (0) support for the size of attributes and parameters to exceed 64 KB. The default message format uses 16-bit integers (max 64 KB) to represent the size of elements in handle value pair sets. Default: 0. For more information, see "9.7. Enabling Support for Large Attributes and Parameters" on page 128.
DM Class Multicast Filtering	
RTI-addDMObjectMulticastAddr	Allows you to assign object classes to separate multicast groups. Default: disabled. For more information, see "9.12. Filtering by Multicast Groups Using Declaration Management" on page 132.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI-addDMIInteractionMulticastAddr	Allows you to assign interaction classes to separate multicast groups. Default: disabled. For more information, see "9.12. Filtering by Multicast Groups Using Declaration Management" on page 132.
RTI Forwarder Parameters	
RTI-addForwarderGroupAddrString	Specifies the IP address of a member of the federate's local Forwarder Group. A separate entry for each RTI Forwarder in the group is necessary. Required only when forwarder groups are used. For more information, see "10.6. Configuring RTI Forwarders for Distributed Forwarding" on page 146
RTI_distributedUdpForwarderMode	Determines whether or not distributed forwarders handle UDP forwarding, and if so, the mode to use. Consistency checking is enforced on this parameter. The mode can be: 0 – Distributed forwarders do not handle UDP forwarding. 1 – Distributed forwarders imitate UDP Forwarders (Legacy Mode). 2 – Distributed forwarders handle combined UDP/TCP forwarding duties. For more information, see "10.6.6 Configuring Distributed UDP Forwarding" on page 152.
RTI_forwarderRoutesFile	Specifies the RTI Forwarder routes file to use. Default: "routes.mtl".  You cannot use this parameter with a centralized RTI Forwarder. For more information, see "5.3. The RTI Forwarder Application" on page 74.
RTI_distributedForwarderPort	Specifies the port for the distributed forwarder. Default: 5000.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_ internalMsgSmartForwardingWhenUsingRtiexec	Enables (1) or disable (0) smart forwarding transport for internal and/or FOM messages (for example, update, reflect, delete, and interactions). Requires the rtiexec process. Ignored if RTI_useRtiExec is not set to 1. All FOM data is sent best effort when FOM reliable is disabled. This should be set to true for most situations as it greatly reduces the number of messages in a federation.
RTI_smartForwardingLevel	Specifies the sender-side filtering mode, where: 0 – no subscription-based routing or filtering. 1 – enable subscription based routing. 2 – enable subscription based routing and disable send for updates and interactions that none of the federates have subscribed to. Default: 0. For more information, see "9.5. Sender-Side Filtering (Smart TCP Forwarding)" on page 124.
RTI_maxNumFederates	Specifies the maximum number of federates allowed in a federation at any one time. This parameter only applies when sender-side filtering is enabled. Each update and interaction message has a bitmask of at least this number of bits appended, so increasing the number of federates increases the overhead. Default: 100. For more information, see "9.5. Sender-Side Filtering (Smart TCP Forwarding)" on page 124.
RTI_maxObjectsPerFederate	Specifies the maximum number of objects that a federate can instantiate. This parameter interacts with the RTI_maxNumFederates parameter.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_forwardingDelay	When publication changes affect forwarding, routing filters are ignored for the specified interval to allow updates to remote subscription information. Default: 3.0.
RTI_routeToAllByDefault	When publication changes affect forwarding, route to all federates until specifically informed by remote federates that they have not discovered an object or are not subscribed to an interaction class. 0 – disable. 1 – enable. Default: 0.
RTI_discoveryMsgBundlingDelay	Delays the sending of discovery messages for the specified number of seconds. This allows discovery messages to be bundled together. Bundling can reduce traffic and processing time if a federate is creating a large number of objects at once. Default: 0.0 seconds.
RTI_enableFedexMsgRouting	This parameter makes sure that internal messages destined for the rtiexec are sent only to the rtiexec and that any messages (both internal and FOM) that the rtiexec does not process are not sent to the rtiexec. Default: 1. 0 – disable filtering. 1 – enable filtering. For more information, see "9.5.2 Configuring Sender-Side Filtering for rtiexec Messages" on page 127.
Save/Restore Configuration	
RTI_enableSaveRestoreWhenUsingRtiexec	Enables (1) or disables (0) save/restore when you are using the rtiexec.

Table 12. Exercise-wide RID file parameters (continued)

Parameter	Description
RTI_saveRestoreTimeout	Specifies the number of seconds the federation execution waits for a save/restore operation to complete before indicating a failure. Default: 120 seconds. For more information, see "6.9. Configuring Save/Restore" on page 97.
RTI_saveTransientMessages	Enables (1) or disables (0) saving of transient messages during a save, to be delivered upon restore. Default: 0. For more information, see "6.9. Configuring Save/Restore" on page 97.
Update Rate Reduction	For details about update rate reduction, see "14.1. Update Rate Reduction" on page 196.
RTI_enableUpdateRateReduction	Enables (1) or disables (0) update rate reduction.
RTI-addUpdateRate	Registers an update rate with the LRC. You must specify a unique name and the update rate in hertz. For example, <code>RTI-addUpdateRate("low", 0.1)</code> registers an update rate called "low" with the value of 0.1 hertz.
RTI-addUpdateRateSubscription	Applies update rate reduction to a specified object class. Can be inclusive or exclusive. (This parameter is federate-specific.)
RTI_receiveTolerance	Specifies a percentage reduction to the inter-arrival time between attribute updates. It reduces the inter-arrival time for a given rate so that an update that appears to arrive early at the receiver is accepted.

A.4. LRC-Specific Parameters

Table 13 lists and briefly describes parameters in *rid.mtl* that can be different for the members of a federation. The table is organized to match *rid.mtl* organization. If applicable, the description includes a cross reference to further information in the manual.

Many of these parameters can be set in the RTI Settings page of the RTI Preferences dialog box, which is the preferred method for editing them. For details, see "4.3. Editing RTI Settings" on page 43.

Table 13. RID file parameters for LRCs

Parameter	Description
RTI_configureConnectionWithRid	Specifies that the rtiexec or federate use the connection configuration parameters in the RID file instead of those in the RTI Assistant connection configurations.
License Checking	
RTI_forceUnlicensedForTwo	Forces the MAK RTI to use the unlicensed version without checking for a license. This setting saves time at startup. If you have a licensed version of the MAK RTI, do not skip license checking. • 0 – Check for a license. • 1 – Do not check for a license. Default: 0.
RTI_disableUnlicensedForTwo	Enables (0) or disables (1) unlicensed mode in Windows (even if the forceUnlicensedForTwo parameter is set). When this parameter is enabled, federates must check out a license to run. Default: 0.
RTI_rtiExecPerformsLicensing	Enables (1) or disables (0) license checkout from the rtiexec. The rtiexec will check out a license for each joining federate. Default 0.
Multicast Discovery Configuration	
RTI_mcastDiscoveryEnabled	Enables (1) or disables (0) multicast discovery. Default: 0. For more information, see "9.4. Automatically Locating the RTI Forwarder and rtiexec" on page 124. This parameter is set to 0 unless RTI_configureConnectionWithRid is set to 1.
RTI_networkInterfaceAddr	Sets the address of the network device to be used for multicast traffic.
RTI_tcpNetworkInterfaceAddr	Specifies the address of the network device for the TCP forwarder.
RTI_mcastDiscoveryPort	Specifies the port for multicast discovery. Default: 6001.
RTI_mcastDiscoveryAddrString	Specifies a multicast discovery address. Default: "229.7.7.8". For more information, see "9.4. Automatically Locating the RTI Forwarder and rtiexec" on page 124.
RTI_lrcMcastDiscoveryDelay	Specifies the interval that the LRC waits for a multicast discovery response. Default: 3.0.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_execMcastDiscoveryDelay	Specifies the interval that the rtiexec waits for a multicast discovery response. Default: 1.0.
RTI_mcastDiscoveryTries	Specifies the number of times the federate should try to find the multicast discovery address before it defaults to best effort transport. Default: 5. For more information, see "9.4. Automatically Locating the RTI Forwarder and rtiexec" on page 124.
Optional Services	
RTI_throwExceptionOnConcurrentAccess	In HLA 1.3, controls whether the ConcurrentAccessAttempted exception is thrown when an RTIambassador service call is called while concurrently processing another service call. This usually happens when a FederateAmbassador callback calls another RTIambassador service call during a tick(). • 0 – Do not throw exceptions. • 1 – Throw exceptions. Default: 0.
RTI_throwExceptionCallNotAllowedFromWithinCallback	If enabled, the HLA Evolved LRC will throw the CallNotAllowedFromWithinCallback when certain RTIambassador service calls are made from a FederateAmbassador callback. (HLA Evolved only)
RTI_throwExceptionForDisabledService	Specifies whether or not function calls throw an exception when they call a service that is not activated in the MAK RTI. • 0 – Do not throw exceptions. • 1 – Throw exceptions. Default: 1. For more information, see "6.8.5 Choosing Silent Failure Instead of Exceptions" on page 96.
RTI_enableWarningsForDisabledServices	Specifies whether or not the RTI Assistant displays a message when an exception is thrown for a disabled service. Default: 1 (on).

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
Switch Table Management	
RTI_preferRidToFedSwitchTable	Specifies which set of switch tables to use. • 1 – Use the RID parameters. • 0 – Use the switch tables defined in the FED file. Force full compliance forces use of the FED file switch table.
RTI_autoProvideSwitch	Specifies whether the Auto Provide switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
RTI_conveyRegionDesignatorSwitch	Specifies whether the Convey Region Designator switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
RTI_conveyProducingFederateSwitch	Specifies whether the Convey Producing Federate switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 1.
RTI_serviceReportingSwitch	Specifies whether the Service Reporting switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
RTI_exceptionReportingSwitch	Specifies whether the Exception Reporting switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
RTI_delaySubscriptionEvaluationSwitch	Specifies whether the Delay Subscription Evaluation switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
Advisory Configuration	
For all parameters in this section, see "4.6. Configuring Use of Advisory Messages" on page 60 for more details.	
RTI_ interactionRelevanceAdvisorySwitch	Specifies whether the Interaction Relevance Advisory switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 1.
RTI_ objectClassRelevanceAdvisorySwitch	Specifies whether the Object Class Relevance Advisory switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 1.
RTI_ attributeRelevanceAdvisorySwitch	Specifies whether the Attribute Relevance Advisory switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
RTI_ attributeScopeAdvisorySwitch	Specifies whether the Attribute Scope Advisory switch is enabled (1), or disabled (0). RTI_preferRidToFedSwitchTable must be enabled to use this parameter. Otherwise it is overridden by the FED file switch table definitions. Default: 0.
Diagnostic Configuration	
RTI_IrcMinidumperLevel	Configures whether the RTI installs hooks to create a dump file if it crashes. Options are: • No Dump. Do not install hooks. • Minidump. Prompt to create a dump file. • Minidump (No Prompt). • Silent Exit. Exit without prompting or creating a dump file. • Default: No Dump.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_notifyLevel	Specifies the category of notification messages to print. 0 – Fatal. 1 – Warn. 2 – Notify. 3 – Verbose. 4 – Debug. Default: 1.
RTI_logFileDirectory	Specifies a directory where diagnostic log files should be created. Default: "" (the federate's working directory).
RTI_logfileName	Enables federate diagnostic logging to a file with the specified name. A file with the specified name is created in the federate's working directory (directory where federate was executed). The same file name is reused and overwritten by all federates with the same working directory unless RTI_reuseLogFile is disabled (0). Default: "" (no file is written). For more information, see "7.3.1 Enabling LRC Diagnostic Data Logging" on page 102.
RTI_rtiExecLogFileName	Enables rtiexec diagnostic logging to a file with the specified name. Default: "" (no file is written). For more information, see "Logging rtiexec Output" on page 67.
RTI_rtiFederationShadowLogFileName	Enables rtiFederationShadow diagnostic logging to a file with the specified name. Default: "" (no file is written)
RTI_rtiForwarderLogFileName	Enables rtiForwarder diagnostic logging to the specified file. Default: "" (no file is written).
RTI_reuseLogFile	Specifies whether diagnostic log file names are reused. When disabled, file names are made unique by appending the application name and system time (in seconds) to the file name. Default: 1 (reuse file names). For more information, see "7.3.1 Enabling LRC Diagnostic Data Logging" on page 102.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_dumpFed	Enables (1) or disables (0) dumping of the contents of the FED file when the notification level is set to verbose (>3). Default: 0.
RTI_dumpRid	Enables (1) or disables (0) dumping of the contents of the RID file when the notification level is set to verbose (>3). Default: 0.
RTI_detachNotifyLevelFromStdOut	Detaches all notification messages at the given level and above from the standard output. This setting has no affect on logging notifications to a file. Default: 5 (no notification levels detached).
RTI_ridConsistencyChecking	Enables (1) or disables (0) checking of the RID file for consistency with the rtiexec's RID file. For more information, see "4.9. The RTI Settings File Consistency Checker" on page 62.
RTI-addRidParametersToOverride	Specifies a list of parameters to add to the default list of parameters that are checked for consistency. For more information, see "4.9.2 Selecting the Parameters to be Checked" on page 63.
RTI-removeRidParametersToOverride	Specifies a list of parameters to remove from the default list of parameters to check for consistency. For more information, see "4.9.2 Selecting the Parameters to be Checked" on page 63.
RTIspy Configuration	
RTI_enableNetworkTesting	Enables (1) or disables (0) network testing.
RTI_enableLRCNetworkStatisticsMonitoring	When enabled (1), a federate's network statistics can be viewed in the RTI Assistant's RTIspy window. This capability is provided by a plug-in. There are limits on how many plug-ins can be active at the same time. When RTI_enableLRCNetworkStatisticsMonitoring is set to 0, all plug-ins should work as expected. When it is set to 1, these constraints are in place (meaning plug-ins will not work or will be completely ignored). • Plug-ins cannot override the DtRtiAmbassadorImplementor. • Plug-ins cannot override the DtConnectionMgr. • Plug-ins cannot override the DtFedAmbWrapper. Default: 1.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
Service Implementation	
RTI_singleCallbackPerTick	Specifies how many packets should be read for each tick. Also applies to tick(min,max). • 0 – A call to RTI::tick with no arguments reads all pending packets. • 1 – A single call to RTI::tick() should result in one packet being processed from the input stream. Default: 0. For more information, see "A.5. Configuring tick (min,max)" on page 233.
RTI_checkFlag	Enables (1) or disables (0) checking the validity of attributes and parameters and the ownership of attributes passed to updateAttributeValues() and sendInteraction(). Default: 1. Setting this flag to 0 results in a small performance gain, but it means that the AttributeNotDefined, AttributeNotOwned, and InteractionParameterNotDefined exceptions will never be thrown by these services.
RTI_processUnknownUpdatesForDiscovery	Enables (1) or disables (0) processing unknown objects for discovery. Default: 1. For more information, see "6.8.4 Processing Discovery of Unknown Objects" on page 96.
RTI_defaultFedFile	Specifies a default FED file in the RID file to support joining without calling create when operating in lightweight mode. In lightweight mode and using the default FED file, the join federation service always associates the federation name with the default FED file. If the value is an empty string, then the FED file will either be established by a previous create federation service call or federation name ".fed". Default: "". For more information, see "6.2. Specifying the FED File" on page 84.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_crcCheckFedFile	If RTI_distributeFedFile is disabled (0), specifies whether or not a federate whose FED file has a different CRC than the rtiexec's FED file can join the federation. 0 – Notify federate if the CRC of its FED file is different from the rtiexec's FED file. Allow the federate to join. 1 – If the CRC of the federate's FED file is different from the rtiexec's FED file, do not allow the federate to join. Default: 0.
RTI_catchFedAmbExceptions	Configures the RTI so that exceptions thrown by the federate in a federate ambassador callback are caught and handled internally by the RTI. If not enabled, the RTI rethrows the exception to be handled by the federate by catching exceptions from tick(). The internal RTI state may be corrupted if this option is disabled. 0 – Do not catch exceptions. 1 – Catch exceptions. Default: 1.
RTI_strictFomChecking	Configures the RTI to strictly check the format of the FDD file (for example, to ensure presence of transport and order or disallow unknown tags). If enabled, an error in the FDD file causes the create federation service to fail. This parameter affects the HLA 1516 or HLA Evolved FDD files. 0 – Do not perform strict checking. 1 – Perform strict checking. Default: 0.
RTI_defaultTimeImplementation	Configures the RTI with the name of a default time implementation if an empty string is provided in the createFederationExecution service call. This default is used to override the default in the logical time library that is returned when the logical time name is an empty string. Default: "".
RTI_flushTimeoutInterval	Specifies the amount of time, in seconds, that an LRC will try to flush messages to the network when it is resigning from a federation. Default: 3.0.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
Advanced Network Configuration	
RTI_mcastTtl	Specifies an integer value that gets used as the TimeToLive parameter in the header of multicast packets. Default: 2.
RTI_enableTcpCompression	Enables compression of RTI message payloads transmitted over TCP connections. Consistency checking is enforced on this parameter. For more information, see "9.6. Compressing RTI Messages" on page 127.
RTI_tcpCompressionLevel	Specifies the amount of compression applied to RTI message payloads over TCP connections: • 1 – minimal compression. • 2 through 8 – increasing levels of compression. • 9 – maximum For more information, see "9.6. Compressing RTI Messages" on page 127.
RTI_enableUdpCompression	Enables compression of RTI message payloads transmitted in UDP datagrams. Consistency checking is enforced on this parameter. For more information, see "9.6. Compressing RTI Messages" on page 127.
RTI_udpCompressionLevel	Specifies the amount of compression applied to RTI message payloads in UDP datagrams, using the same options as RTI_tcpCompressionLevel . For more information, see "9.6. Compressing RTI Messages" on page 127.
RTI_tcpNoDelay	Allows you to configure the RTI to emphasize minimal latency or maximum throughput. • 0 – Trade latency for throughput. • 1 – Set the TCP_NODELAY socket option for TCP sockets, yielding minimum latency. Default: 1.
RTI_tcpBufferSize	Specifies the size of the buffer into which TCP packets get read. Default: 500000 bytes. For more information, see "9.1. Choosing Best Effort or Reliable Transport" on page 122.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_socketReceiveBufferSize	Specifies the size of the TCP and UDP network receive buffers. Increasing the size can increase the RTI's tolerance of peak network loading. Default: 2000000. For more information, see "9.8. Configuring the Network Buffer Sizes" on page 129.
RTI_socketSendBufferSize	Specifies the socket send buffer size. Default: 200000.
RTI_enablePacketBundling	Bundles messages together into a single TCP packet or UDP datagram, up to the size limit specified by RTI_packetBundlingSize . Default: 1400. For more information, see "9.3. Packet Bundling for Reliable and Best Effort Messages" on page 123.
RTI_packetBundlingSize	When packet bundling is enabled, specifies the size of the message bundles for TCP packets and UDP datagrams. The value should be $\leq$ MTU of the network. Default: 0. For more information, see "9.3. Packet Bundling for Reliable and Best Effort Messages" on page 123.
RTI_tcpPacketBundlingSize	When packet bundling is enabled, specifies the size of the message bundles for TCP packets. Default: 50000.
RTI_udpPacketBundlingSize	When packet bundling is enabled, specifies the size of the message bundles for UDP datagrams. The value should be $\leq$ MTU of the network. Default: 1400.
RTI_waitIOPeriodForBundling	Enable RTI_waitIOPeriodForBundling when you want the LRC to wait at most the IO period (RTI_IOPeriod) before sending data if no other messages are being queued to fill a bundle. When disabled this prevents the latency overhead of waiting the maximum amount of time for more messages to completely fill a bundle. However, it also may cause messages not to be bundled if the throughput is low.
RTI_maxUdpPacketSize	Specifies the maximum size for a UDP packet. The default is 15000 for legacy purposes. However, packets this large are not recommended.
RTI_enableBestEffortSendRetry	Enables (1) or disables (0) automatic resending of failed attempts to send best effort messages. Configured with RTI_bestEffortSendRetryWaitUsec and RTI_bestEffortSendRetries . Default 1.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_bestEffortSendRetries	If RTI_enableBestEffortSendRetry is enabled (1), RTI_bestEffortSendRetries configures the number of times a failed best effort send is retried. Default: 50. For more information, see "9.9. Controlling Congestion for Best Effort Sockets" on page 130.
RTI_bestEffortSendRetryWaitUsec	If RTI_enableBestEffortSendRetry is enabled, RTI_bestEffortSendRetryWaitUsec configures the time to wait between failed best effort sends, in microseconds. Default: 50.
RTI_useBusyWaitForTickMinMax	Specifies that the RTI will use busy wait in tick (min,max) when no messages are pending. • 0 – Disabled (use select). • 1 – Enabled. Default: 0. For more information, see "A.5. Configuring tick (min,max)" on page 233.
RTI_enableMessageThrottling	Enables (1), or disables (0) throttling of transmission and receipt of internal messages (for all versions of the RTI). This parameter also enables or disables buffering of provideAttributeValueUpdate callbacks in HLA 1516 or HLA Evolved. Default: 0 (disabled).
RTI_transmitDelay	If RTI_enableMessageThrottling is enabled this parameter configures the time to delay before processing requests. Default 0. For more information, see "11.2.1 Configuring the Delay Time for Processing Internal Messages" on page 163.
RTI_transmitRate	If RTI_enableMessageThrottling is set to 1, RTI_transmitRate specifies the number of messages per second that you want to send. Default 0.0 (disabled). For more information, see "11.2.2 Transmitting Internal Messages Across Ticks" on page 164.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI_autoProvideDelay	If RTI_enableMessageThrottling is set to 1, enables buffering of automatic provideAttributeValueUpdate callbacks in HLA 1516 or HLA Evolved to limit the number of provides requested if multiple federates require updates. Default: -1 (disabled). For more information, see "9.11. Buffering Automatic Update Requests (1516 Specifications Only)" on page 131.
RTI_dualTransmitFirstInteractionRegions	Enables (1) or disables (0) sending the DDM interaction region information both best effort and reliably the first time. A race condition between a best effort DDM interaction and its region information (sent reliably) is alleviated by transmitting the region information via best effort the first time. Default: 0.
RTI-addDestAddrString	Adds one or more additional destination addresses for best effort packets, which can be broadcast, unicast, or multicast addresses. All best effort data is sent to all registered addresses. Default: "0.0.0.0". For more information, see "10.2.2 Configuring UDP Over a WAN Without Distributed Forwarding" on page 138.
RTI_maxForwarderQueue	Specifies the size of the queue for messages that exceed the size of the receive buffer. When a federate does not tick fast enough and its receive buffer fills, it starts placing the messages into the queue. Once the queue has reached the maximum size, the federate is dropped. If the size is set to -1 (default), it queues messages indefinitely until the federate processes them all, or until all system memory is completely consumed.
RTI_allProvideUpdateRequestsDelayed	Specifies that RTI_autoProvideDelay should apply to all update requests, not just those generated due to the autoProvide switch. Message throttling (RTI_enableMessageThrottling) must be enabled and RTI_autoProvideDelay must be set to a non-zero value.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
Asynchronous Processing	
RTI_processingModel	Specifies the synchronous/asynchronous I/O operating mode: • 0 – Synchronous mode. • 1 – Asynchronous I/O. Perform network reads and writes in an asynchronous thread (reduces dependency on tick rate to avoid message drops). • 2 – Asynchronous Message Processing. In addition to performing network reads, the asynchronous thread processes messages. However, it will not perform any asynchronous federate ambassador callbacks. The callbacks will be cached and delivered when the federate calls tick or evokeCallbacks. • 3 – Asynchronous Callbacks. Process messages and invoke callbacks in an asynchronous thread.  For Asynchronous Callbacks (not Asynchronous I/O), the Federate must be able to receive Federate Ambassador callbacks asynchronously (in a thread safe way). Default: 1.
RTI_tickWaitPeriod	Determines the smallest increment of time that a tick will cause a federate to wait for messages from the I/O thread when using asynchronous I/O. If set to 0.0, the minimum time specified in the tick service invocation is used instead. Default 0.0.
RTI_IOPeriod	Specifies the amount of time polling operations take, in seconds. Not used in Windows, which is event driven. Must be greater than or equal to zero. Default: 0.1.
RTI_tickFavorsNetwork	Enables (1) or disables (0) a thread yield in tick(). Default: 1.
RTI_maxIOQueue	Specifies how many messages can be queued to send before blocking. Default 500000.
RTI_maxIOCount	Specifies the number of times the thread can cycle before it must yield. Must be greater than zero. Default: 50000.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
DDM Configuration	
RTI_minChannelAddr	Specifies the starting network multicast address for a pool of multicast groups used by DDM. The pool defined by this parameter and the RTI_maxChannelAddr parameter is used by any federate that loads this configuration file. Default: "224.0.1.0" if using IPv4; "ff12::3:1" if using IPv6.
RTI_maxChannelAddr	Specifies the final network multicast address for a pool of multicast groups used by DDM. The pool defined by this parameter and the RTI_minChannelAddr parameter is used by any federate that loads this configuration file. Default: "239.255.255.255" if using IPv4; "ff12::3:fff" if using IPv6. For more information, see "12.1.3 Configuring a Fixed Grid DDM" on page 169.
RTI_addressDelay	Specifies the delay by which data associated with changed regions and associations is broadcast (default multicast) while the DDM routing information is propagated to remote federates. Default: 1.0.
RTI_conveyOnlyAvailableDimensions	If RTI_conveyOnlyAvailableDimensions is enabled (1), the conveyed regions only include those dimensions that are available to the received interaction class. If the regions used to send the interaction include dimensions that are not available at the received interaction class, they are removed from the conveyed regions. If RTI_conveyOnlyAvailableDimensions is disabled (0), the conveyed regions include exactly those dimensions that were in the regions used to send the interaction. As a result, there may be dimensions that are not available at the received interaction class. Default: 0.
RTI_implicitDdmParamsFile	Specifies the name of the file that has implicit DDM configuration parameters. Default: <i>implicitDdmParams.mtl</i> .

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
MOM Configuration	
For details about all parameters in this section, see "11.1.1 Configuring MOM Services" on page 160.	
RTI_momVerboseLevel	It controls the verbosity level of MOM service reporting. Specify 0 or a non zero. When non zero, the service reporting makes strings from things like class and attribute handle using the FOM names versus just the handle "toString" representation which would just be a numeric value. So a register object instance (object class handle) service call report would be something like "registerObjectInstance {\"HLAObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft\"}" with verbose versus registerObjectInstance {28} without verbose.
RTI_momExceptionLogging	Enables (1) or disables (0) MOM exception logging. Default: 0.
RTI_momFederateUpdateInterval	Specifies the default interval in seconds at which the MOM Federate object will be updated. Default: 0 (no updates sent).
Save/Restore Configuration	
RTI_saveRestoreDirectory	Specifies the directory used for writing and reading save files. If the saved federation contains multiple federates of the same type, each federate of a given type must have access to all of the LRC save files of that type. The RTI determines the save files used for each federate. Default: ".".
Modular FOMs	
For details about modular FOMs, see "13.1. Using FOM Modules" on page 188.	
RTI_fomModuleMerging	Enables (1) or disables (0) merging of FOM modules. Default: 0.
RTI_fomModuleSorting	Enables (1) or disables (0) FOM module sorting when merging. FOM module sorting sorts the set of FOM module designators passed to the Create Federation Execution and Join Federation Execution services based on the Dependency value in the HLA1516-2010 FOM file's modelIdentification element. Without this, the RTI merging assumes that the FOM designators are ordered with respect to scaffolding and other OMT dependencies.

Table 13. RID file parameters for LRCs (continued)

Parameter	Description
RTI-addCreateFomModule	Specifies a FOM module to add when creating a federation.
RTI-addJoinFomModule	Specifies a FOM module file to add when joining a federation.
RTI_momModuleExtensionFileName	Specifies that the LRC will use the local FOM module file rather than having the rtiexec send it back.
RTI_preferLocalFomModule	Specifies a MOM module file to add when creating a federation.

A.5. Configuring tick(min,max)

By default, the implementation of tick(min,max) uses select (or windows events) to wait for additional input when there are no pending messages. The granularity of the select mechanism can make the tick(min,max) exceed the maximum time by too much (for example, select takes approximately 10 msec under Linux). To have finer control of when tick(min,max) returns, use the **RTI_useBusyWaitForTickMinMax** parameter to replace the select mechanism with a busy wait that polls the input source.

 We do not recommend enabling the busy wait mechanism if you use asynchronous I/O. The busy wait will not yield to the I/O thread and the I/O thread must execute to generate input.

The **RTI_singleCallbackPerTick** parameter also applies to tick(min,max). When **RTI_singleCallbackPerTick** is enabled, tick(min,max) returns after processing a single message or when the minimum time has been exceeded (effectively max is ignored).

 A message may only cause internal processing and not necessarily generate a callback. The return value of tick() or tick(min,max) will be true if a message has been processed; otherwise, it will be false.



B. Example Federates

This appendix describes the example federates provided with the MAK RTI.

B.1. Example Federates	236
B.2. Building the Examples	236
B.3. The rtisimple Example	236
B.4. The simpleDDM Example	237
B.5. The simpletime Example	238
B.6. HLA Bounce	240
B.6.1 Running HLA Bounce	240
B.6.2 Demonstrating Ownership Management	241
B.6.3 Demonstrating DDM	242
B.7. The MAK Shooter Example	242
B.7.1 Building MAK Shooter	243
B.7.2 Running MAK Shooter	243
B.7.3 Playing MAK Shooter	244

B.1. Example Federates

Explaining how to develop federates for use with HLA is beyond the scope of this manual. In general, when customers ask us how to develop federates, we recommend using the protocol-independent interface provided by VR-Link. For customers who just need a federate to do some basic exploration of HLA and the MAK RTI, we recommend using the example applications provided with VR-Link. However, for those customers who want to develop directly to the HLA API, we include the simple examples described in this appendix.

The examples are in the `./examples` directory. Compiled versions of each example are in `./bin`. Java examples put the `.jar` files in their respective examples directories with the run scripts. Each example has an HLA 1.3 version, an HLA 1516 version, an HLA Evolved version, and an HLA 4 (1516_2025) version except the Java MakShooter example, which is available only for HLA 1516 Evolved, and the Simple Federate Protocol example, which is available only for HLA 4.

This appendix provides brief introductions to the examples. For complete descriptions, please see the Examples link in the online API documentation.

B.2. Building the Examples

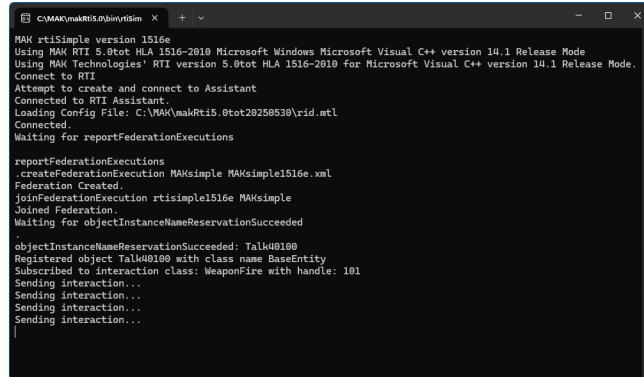
To build the C++ examples, use the Visual Studio solutions files (for Windows) or Makefiles (for Linux) in `./rti/examples`. Compiling those projects will create the examples.

Certain examples, such as HLA Bounce and simpleDDM, have a dependency on Qt. To build them, you must set the environment variable `QTDIR` to point to your installation of Qt. You can download Qt from our website: <https://www.mak.com/mak-one/support/help/bonus-material>

B.3. The rtisimple Example

The `rtisimple` example is a simple C++ example of how to use the RTI API. The `rtisimple` federate tries to create a federation and join it. It publishes and subscribes to the `BaseEntity` object class and registers an object of that class. Then it begins to update the object attributes and periodically sends an interaction. The attribute and parameter values contain the string names of the attributes and parameters. If two or more `rtisimple` federates are running, each federate discovers the other's objects, reflects the other's updated attributes, and receives the other's interactions as shown in Figure 63 and Figure 64.

Figure 63. rtisimple console sending interaction



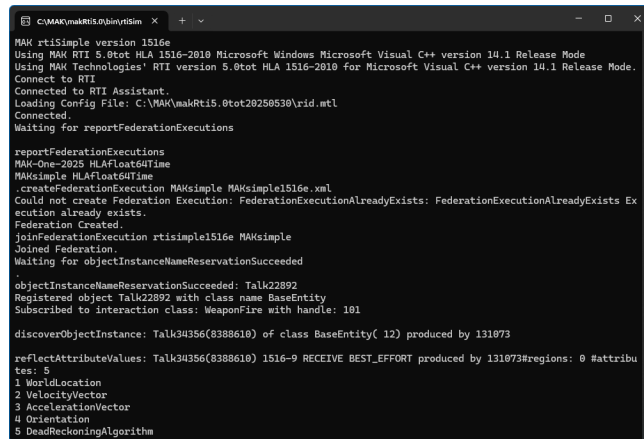
```

C:\MAK\makRTI5.0\bin\rtisim X
MAK rtisimple version 1516e
Using MAK RTI 5.0tot HLA 1516-2010 Microsoft Windows Microsoft Visual C++ version 14.1 Release Mode
Using MAK Technologies' RTI version 5.0tot HLA 1516-2010 for Microsoft Visual C++ version 14.1 Release Mode.
Connect to RTI
Attempt to create and connect to Assistant
Connected to RTI Assistant.
Loading Config File: C:\MAK\makRti5.0tot20250530\rid.mtl
Connected.
Waiting for reportFederationExecutions

reportFederationExecutions
.createFederationExecution MAKsimple MAKsimple1516e.xml
Federation Created.
joinFederationExecution rtisimple1516e MAKsimple
Joined Federation.
Waiting for objectInstanceNameReservationSucceeded
objectInstanceNameReservationSucceeded: Talk40100
Registered object Talk40100 with class name BaseEntity
Subscribed to interaction class: WeaponFire with handle: 101
Sending interaction...
Sending interaction...
Sending interaction...
Sending interaction...

```

Figure 64. rtisimple console receiving interaction



```

C:\MAK\makRTI5.0\bin\rtisim X
MAK rtisimple version 1516e
Using MAK RTI 5.0tot HLA 1516-2010 Microsoft Windows Microsoft Visual C++ version 14.1 Release Mode
Using MAK Technologies' RTI version 5.0tot HLA 1516-2010 for Microsoft Visual C++ version 14.1 Release Mode.
Connect to RTI
Connected to RTI Assistant.
Loading Config File: C:\MAK\makRti5.0tot20250530\rid.mtl
Connected.
Waiting for reportFederationExecutions

reportFederationExecutions
MAK-One-2025 HLAFloat64Time
MAKsimple HLAFloat64Time
.createFederationExecution MAKsimple MAKsimple1516e.xml
Could not create Federation Execution: FederationExecutionAlreadyExists: FederationExecutionAlreadyExists Ex
ecution already exists.
Federation Created.
joinFederationExecution rtisimple1516e MAKsimple
Joined Federation.
Waiting for objectInstanceNameReservationSucceeded
objectInstanceNameReservationSucceeded: Talk22892
Registered object Talk22892 with class name BaseEntity
Subscribed to interaction class: WeaponFire with handle: 101

discoverObjectInstance: Talk34356(8388610) of class BaseEntity( 12) produced by 131073

reflectAttributeValues: Talk34356(8388610) 1516-9 RECEIVE BEST_EFFORT produced by 131073#regions: 0 #attribu
tes: 5
1 WorldLocation
2 VelocityVector
3 AccelerationVector
4 Orientation
5 DeadReckoningAlgorithma

```

The rtisimple federates for HLA 1.3, 1516, HLA Evolved, and HLA 4 can interoperate using the MAK RTI. They all use a narrow string format for the attribute values and tags. The default federation file format matches the respective federate:

- HLA 1.3 – rtisimple13 uses *MAKsimple.fed*.
- HLA 1516 – rtisimple1516 uses *MAKsimple.xml*.
- HLA Evolved – rtisimple1516e use *MAKsimple1516e.xml*.
- HLA 4 – rtiSimple1516_2025 uses *MAKsimple1516_2025.xml*.

The MAK RTI supports both formats for the different APIs. However, the MOM specification is different for 1.3 and 1516. If the MOM is enabled in the RID file, then the appropriate federation file must be used.

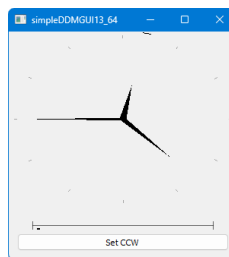
B.4. The simpleDDM Example

The simpleDDM example is an adaptation of the rtisimple example program that incorporates DDM. (For a definition of DDM, please see "[Data Distribution Management](#)" in the *MAK RTI User Guide*. For details about how the MAK RTI implements DDM, see "Chapter 12. Using Data Distribution Management" on page 167.) It performs the same typical federation calls as rtisimple (join, create, and so on) with the exception that when an object is registered, it is done so "with Region."

The example has an object attribute and an interaction that is published and subscribed with region. To see the affects and features of the example, you must create two federates. One federate must be a publisher (`-isPublisher 1`) and the other a subscriber (`-isPublisher 0`). The regions in the example are conceptualized in one dimension as seconds along a 12 hour axis (ranging from 0 seconds to 43200) and in another dimension as a global time zone.

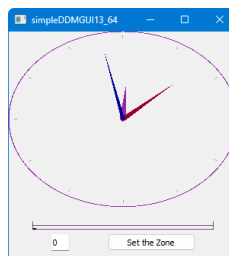
The published object represents the current time of the publisher and the published interaction represents an hourly chime. One chime (quarterlyChime) occurs at 12, 3, 6 and 9 o'clock; another chime occurs at the other hours. The `timeZone` dimension ranges from 0 to 23. It represents the different time zones around the world. It can only be changed in the publisher on the command line (`-timeZone time_zone`). You can change it dynamically in the subscriber GUI.

Figure 65. simpleDDM publishing federate



The GUI displays clock hands, which for the publishing federate show the current time (Figure 65), and for the subscribing federate show the last reflected position of the publisher (Figure 66). (The subscriber's hands are not displayed prior to object discovery.) The GUI displays (as an arced segment around the dial) the region in the seconds dimension. It also displays a horizontal timeline at the bottom of the dial representing the 12 hour extent, a horizontal line representing the subscribed region, and the current position or last reflected position.

Figure 66. simpleDDM subscribing federate



To specify the subscribed region in the subscribing federate, use the command line options `-minSubscribe hour` and `-maxSubscribe hour`. You can specify the region in seconds using `-minSubscribe seconds` and `-maxSubscribe seconds`.

B.5. The simpletime Example

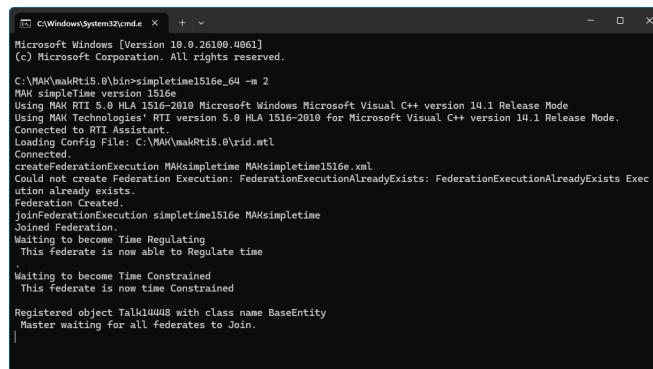
The simpletime example is a simple C++ example of how to use the time management features of the RTI API. The simpletime federate tries to create a federation and join it. It enables time management in the RTI by becoming both time managed and time

constrained. It publishes and subscribes to the BaseEntity object class and registers an object of that class. Using a federation synchronization point, it synchronizes the federation. Then it publishes and subscribes to the WeaponFire and MunitionDetonation interaction classes.

After this initialization process, the federate begins to increase a stored time and position value, updates its attribute values accordingly, and if appropriate, sends a fire or detonate interaction. The federate sends the fire interaction when its stored position exceeds a preset value (the phase-line), or you issue a keyboard command instructing the federate to fire. Then a detonation is sent. If no detonations are sent, the federate continues to update its attribute values and increase its time, while regularly printing out its time state information and its stored position.

If you run two or more simletime federates, each federate discovers the other's object, reflects the other's updated attributes, and receives the other's interactions, as shown in Figure 67 and Figure 68.

Figure 67. simletime federate started with -m 2



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26100.4861]
(c) Microsoft Corporation. All rights reserved.

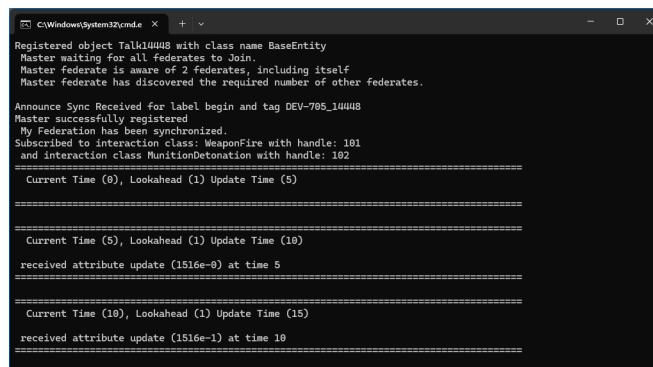
C:\MAK\makRti5.0\bin>simletime1516e_64 -m 2
MAK simletime version 1516e
Using MAK RTI 5.0 HLA 1516-2010 Microsoft Windows Microsoft Visual C++ version 14.1 Release Mode
Using MAK Technologies' RTI version 5.0 HLA 1516-2010 for Microsoft Visual C++ version 14.1 Release Mode.
Connected to RTI Assistant.
Loading Config File: C:\MAK\makRti5.0\rid.mtl
Connected.
createFederationExecution MAKsimletime MAKsimletime1516e.xml
Could not create Federation Execution: FederationExecutionAlreadyExists: FederationExecutionAlreadyExists Exec
ution already exists.
Federation Created.
joinFederationExecution simletime1516e MAKsimletime
Joined Federation.
Waiting to become Time Regulating
This federate is now able to Regulate time

Waiting to become Time Constrained
This federate is now time Constrained

Registered object Talk10008 with class name BaseEntity
Master waiting for all federates to Join.

```

Figure 68. simletime federate after second federate starts



```

C:\Windows\System32\cmd.exe
Registered object Talk10008 with class name BaseEntity
Master waiting for all federates to Join.
Master federate is aware of 2 federates, including itself
Master federate has discovered the required number of other federates.

Announce Sync Received for label begin and tag DEV-705_10008
Master successfully registered
My Federation has been synchronized.
Subscribed to interaction class: WeaponFire with handle: 101
and interaction class MunitionDetonation with handle: 102

=====
Current Time (0), Lookahead (1) Update Time (5)

=====

Current Time (5), Lookahead (1) Update Time (10)
received attribute update (1516e-0) at time 5

=====

Current Time (10), Lookahead (1) Update Time (15)
received attribute update (1516e-1) at time 10

=====

```

Start the first simletime federate with the command line option `-m totalNumberOfFederates`. This tells the federate to wait until the specified number of federates (including itself) have joined the federation before synchronizing the federation and allowing the other federates to run. You can start subsequent federates without any command line options.

B.6. HLA Bounce

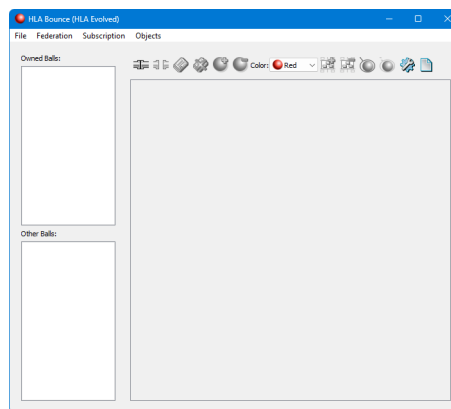
The HLA Bounce example is an HLA federate that demonstrates subscription/unsubscription, ownership management, and DDM in the context of a whimsical graphical user interface. Each instance of HLA Bounce displays one or more bouncing balls. You can change their color, view balls controlled by other instances of HLA Bounce, change the ownership of the bouncing balls, and view only those balls that are in a specified region.

B.6.1 Running HLA Bounce

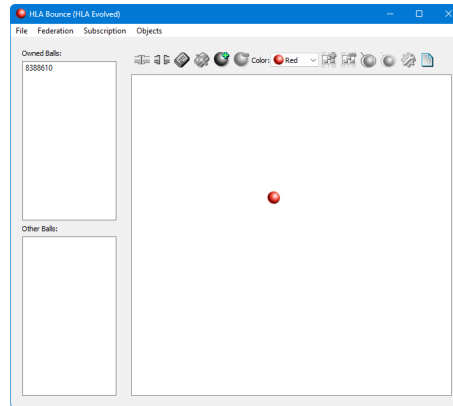
► To run HLA Bounce:

1. On the Windows Start menu, choose **MAK Technologies > MAK RTI 5.0 Examples**. In the Examples folder, double-click one of the shortcuts **HLA Bounce version**, where *version* is 1.3, 1516, 1516e, or 1516_2025. The Choose RTI Connection dialog box opens.
2. Select the connection you want to use. To use ownership management or DDM, run the `rtiexec`.
3. Click Connect. HLA Bounce opens (Figure 69).

Figure 69. HLA Bounce



4. Start a second instance of the HLA Bounce example.
5. Select one of the two HLA Bounce windows.
6. Choose **Federation Create/Join**.
7. Choose **Objects > Add Ball**. A red ball (the default color) is added to the window and starts bouncing around (Figure 70).

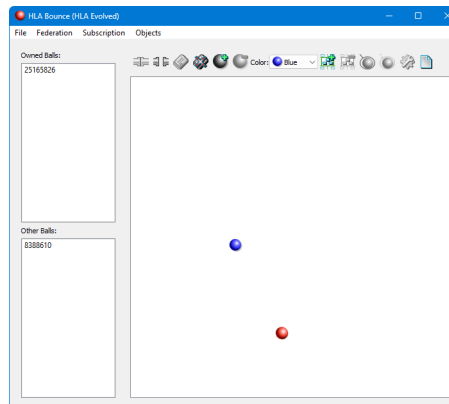
Figure 70. Bouncing ball

8. Select the other HLA Bounce window.
9. Choose Federation Create/Join.
10. In the Federate Color drop-down list, choose a color other than red.
11. Add a ball.

Both windows now have balls bouncing in them, but they only show their own balls.

► In one of the two HLA Bounce windows, choose **Subscription > Subscribe**.

The window now shows the local ball and the ball being simulated by the other federate (Figure 71). The locally simulated ball is listed in Owned Balls. The remotely simulated ball is listed in Other Balls.

Figure 71. HLA Bounce subscription enabled

B.6.2 Demonstrating Ownership Management

► To demonstrate ownership management:

1. Start two HLA Bounce windows. Use an rtiexec connection and force full compliance.
2. Start one or more balls in each window.
3. In one of the windows, select an entry in Other Balls.

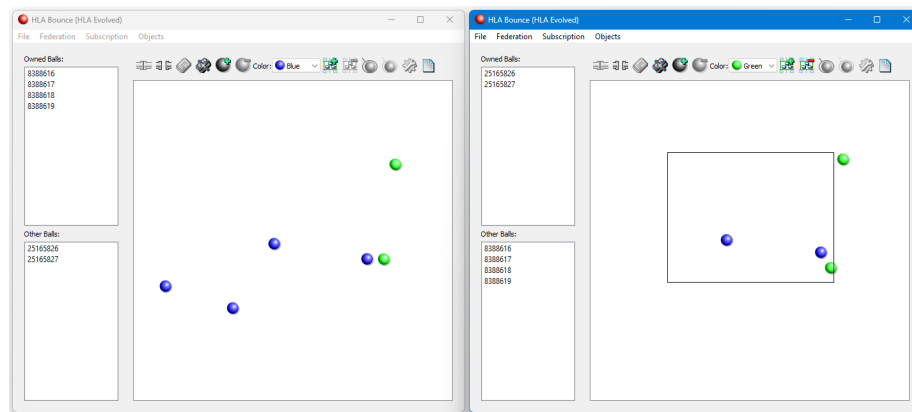
4. Choose **Objects > Acquire Ball**. The entry for the remote ball is now in the Owned Balls list and the ball changes color to match the color in this instance of HLA Bounce.

B.6.3 Demonstrating DDM

► **To demonstrate DDM:**

1. Start two HLA Bounce windows. Use an rtiexec connection and force full compliance.
2. Start one or more balls in each window.
3. In each window choose **Subscription > Subscribe**.
4. In one of the windows, Choose **Subscription > Add Region**. The cursor changes to a cross-hair.
5. In the window, draw a rectangular area.
6. Watch the two windows and see where the balls are bouncing. In the window in which you drew a region, you see the remote balls only when they are within the region. (You see them all the time in their home window.) In Figure 72, although the remote federate is simulating four blue balls (as shown in the Other Balls list), only the two within the subscription region are visible. (The federate is not designed to undiscover balls when they leave the region, so they remain listed in the Other Balls list.)

Figure 72. Subscription with region



B.7. The MAK Shooter Example

The MAK Shooter example is an HLA federate that demonstrates object publishing, subscribing, updating of attributes, and sending of interactions in the form of a simple arcade style shooter game. Each instance of MAK Shooter creates a player's screen, on which you can navigate a ship to fire and dodge shots. In addition, you can join a game as an observer (a federate which only subscribes to Ship and Shot objects, not publish) and observe games in action.

MAK Shooter is a Java application and to build it you must install the JDK. For complete details, see *./makRti5.0/examples/java/MakShooter/README.txt*.

B.7.1 Building MAK Shooter

To build MAK Shooter, navigate to `./makRti5.0/examples/java/MakShooter` and run the appropriate build script (*buildInWindows.bat* for Windows and *buildInLinux.sh* for Linux). This generates *MakShooter.jar*.

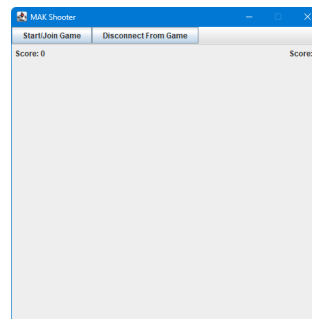
After building the game, you can play it.

B.7.2 Running MAK Shooter

► To run MAK Shooter:

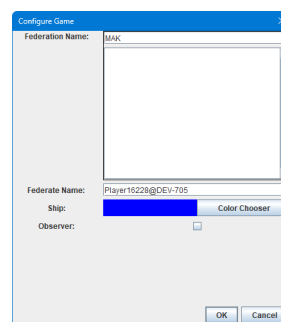
1. Navigate to `./makRti5.0/examples/java/MakShooter` and run the appropriate run script (*makShooter1516e.bat* in Windows and *makShooter1516e.sh* in Linux). The game opens (Figure 73).

Figure 73. MAK Shooter



2. Click the Start/Join Game button.
3. Select the Connection you want to use.
4. Click Connect. The Configure Game dialog box opens (Figure 74).

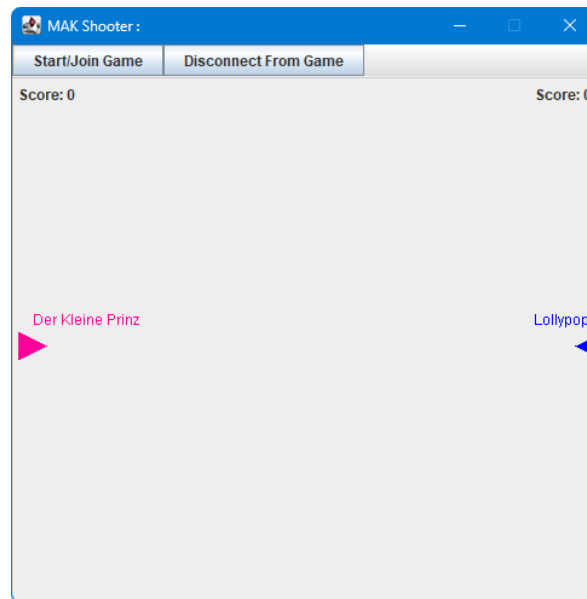
Figure 74. Configure Game dialog box



5. Fill in the fields as necessary:
 - a. Federation Name: The name of the federation. If you are joining a game, select a federation from the list. If starting a new game, you must enter a unique federation name.
 - b. Federate Name: Enter the name of your ship.

- c. Ship: The color of your ship. By default, the ship is blue. To change the color, click the Color Chooser button and select the desired color.
 - d. Observer: If there is a game already in progress, and you want to observe, select this option and join a game as an observer. Observers only watch the game and cannot control a ship. The game only becomes active when two people have started MAK Shooter.
6. Click OK.
 7. Start a second instance of MAK Shooter by following steps 1 through 5. Be sure to use the same Federation name and a different Federate Name.
 8. Both windows become active with the ships at each end (Figure 75).

Figure 75. Game started

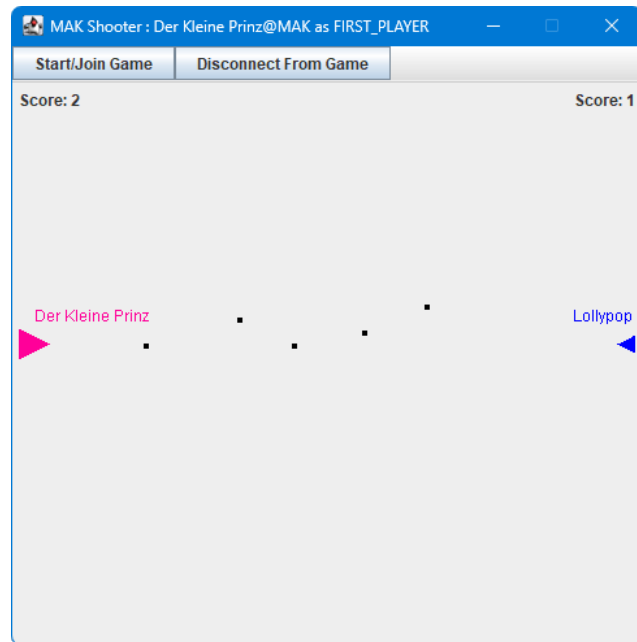


9. Play the game as described below.

B.7.3 Playing MAK Shooter

► To play the game:

- Use the Up and Down Arrow keys to move the ship.
- Use the Space Bar key to fire. If a shot hits the opposing ship, a detonation interaction occurs, the Score increments, and the ship flashes from the interaction.

Figure 76. Shots fired

- To quit the game, click the Disconnect From Game button. The other player will be notified and disconnected.



Index



- command-line option, 74, 77-78, 80
- A command-line option, 76, 79
- a command-line option, 78



- adding
 - RID file, 57
- address
 - destination, 207
 - best effort, 229
 - IP, 136
 - multicast
 - final, 231
 - starting, 231
 - multicast discovery, 218
 - network interface, 218
 - RTI Assistant, 73
 - sending to multiple, 138
 - TCP Forwarder, 207



- address command-line option, 78
- advisory message
 - discovered class, 210
 - sending, 60
- All Provides Delayed RTI setting, 131
- ancillary forwarder, 140

API

- RTIsPy, 22
- application, multi-threaded, 26
- asynchronous
 - callbacks, 26-27, 29
 - message processing, 28
 - processing, 27, 130
- asynchronous callback, 54, 230
- asynchronous I/O, 28
 - concepts, 26
- asynchronous IO, 54, 230
- asynchronous message processing, 54, 230
- attribute
 - passelization, 174
 - sending, 122
 - large, 128
 - validity checking, 224
- attribute ownership unavailable callback, 50, 211
- Attribute Relevance Advisory
 - setting, 51
 - switch, 51
- Attribute Relevance Advisory switch, 221
- Attribute Scope Advisory
 - setting, 51
 - switch, 51
- Attribute Scope Advisory switch, 221
- Auto Provide Delay setting, 47
- auto provide switch, 50, 220



- autoExit command-line option, 73
- Automatically Delete Orphans
 - setting, 94

Automatically Delete Orphans setting, 47, 92
autoProvide, 48
Auto-Provide setting, 50

B

bandwidth
 minimizing use of, 124
 reducing, 123
best effort, 29, 136, See also UDP
 choosing, 122
 configuring for LAN, 136
 destination addresses, 229
 over WAN, 138
 topologies, 29
 unicast distributed forwarding, 152
 WAN, 30
best effort message, 227
binding
 Java, 40
broadcast, 29-30
 address, 136
 RTI Assistant, 73
broken connection, 94
buffer
 receive, 129
 send, 129
 socket receive, 55, 227
 socket send, 56, 227
 TCP, 55, 129, 226
buffering
 update requests, 131
building
 examples, 236
 with 1.3, 35
 with 1516, 36
bundle
 size, 148
bundling
 enabling, 148
 message, 47, 216
 packet, 56, 227
 packets, 123

-

-C command-line option, 73

C

callback
 asynchronous, 26-27, 29, 54, 230
 attribute ownership unavailable, 50, 211
 FederateAmbassador, 219
CallNotAllowedFromWithinCallback exception,
 219
centralized TCP forwarding, 30
 configuring, 136
 on WAN, 137
checking
 attribute validity, 224
 FOM, 50, 225
 license, 57, 218
 RID file consistency, 62
checkout, license, 57, 218
choosing transport type, 122
class
 HLAfloat64Interval, 40
 HLAfloat64Time, 40
 HLAfloat64TimeFactory, 40
 HLAinteger64Interval, 40
 HLAinteger64Time, 40
 HLAinteger64TimeFactory, 40
 LogicalTimeFactoryFactory, 39-40
 LogicalTimeFactoryImpl, 39
 LogicalTimeFactoryImplFloat, 39
 LogicalTimeFactoryImplInteger, 39
 LogicalTimeImpl, 39
 LogicalTimeImplFloat, 39
 LogicalTimeImplInteger, 39
 LogicalTimeInterval, 38
 LogicalTimeIntervalImpl, 39
 LogicalTimeIntervalImplFloat, 39
 LogicalTimeIntervalImplInteger, 39
 RTIfedTime, 38
clearing
 console or log window, 72
closing
 RTI Assistant, 69
 RTI Forwarder, 77
command
 delete, 82
command-line option
 --, 74, 77-78, 80
 -A, 76, 79
 -a, 78
 --address, 78
 --autoExit, 73
 -C, 73

- Connection, 73
- d, 74, 76
- D, 76
- destAddrString, 76, 79
- disableMulticast, 74
- distributedForwarderPort, 76
- doNotUseRoutesFile, 76
- F, 74, 76, 79
- f, 76, 79
- federationHandle, 79
- federationName, 79
- Federations, 74
- forceFullCompliance, 76
- forceShutdown, 74
- forwarderPortToConnectTo, 76
- forwarderToConnectTo, 76
- H, 74
- h, 74, 77-78, 80
- help, 74, 77-78, 80
- History, 74
- i, 80
- ignore_rest, 74, 77-78, 80
- K, 73
- l, 77, 79
- M, 76, 79
- manual, 76, 79
- n, 74, 76, 78-79
- N, 76, 80
- networkInterfaceAddr, 76
- notifyLevel, 74, 76, 78-79
- P, 77, 80
- p, 77-78, 80
- port, 78
- q, 79
- r, 77, 80
- R, 77, 79
- routesFileName, 77
- RTI Forwarder, 76
- S, 74
- s, 74
- setLogFileName, 77, 79
- setNotifyQuiet, 79
- setRidFileName, 77, 79
- setTcpPort, 77, 80
- setUdpPort, 77, 80
- shutdownRequest, 74
- T, 77, 80
- t, 77, 80
- tcpAddress, 77, 80
- tcpNetworkInterfaceAddr, 80
- udpNetworkInterfaceAddr, 80
- usePreconfiguredRid, 77, 80
- useReliable, 80
- v, 74, 77-78, 80
- version, 74, 77-78, 80
- command-line option, ignore rest, 74, 78, 80
- command-line options
 - RTI Assistant, 73
 - RTI Federate Protocol Server, 78
 - RTI Federation Shadow, 79
- comment
 - MTL parameter, 204
- communicating
 - over WAN, 138
- compatibility
 - link, 23
 - network, 22
 - with other RTIs, 20
- compiling
 - against MAK RTI, 34
 - for 1.3, 35
 - for 1516, 36
 - on Windows, 35-36, 38
- compliance
 - full, 206
 - HLA, 160
- compliant mode
 - specifying, 76
- compressing RTI messages, 127
- compression
 - enabling, 148
 - level, 148
 - TCP, 56, 226
 - TCP and UDP, 63
 - UDP, 57, 226
- ConcurrentAccessAttempted exception, 219
- configuration
 - display settings, 59
- configuration file
 - fixed grid, 209
 - RID, 22
- configuring
 - asynchronous I/O, 28
 - best effort on LAN, 136
 - centralized TCP forwarding, 136
 - on WAN, 137
 - create FOM module, 192
 - DDM
 - fixed grid, 169
 - declaration management, 132
 - delay time for internal messages, 163
 - diagnostics, 67
 - discovery processing, 96
 - distributed forwarding, 146
 - hierarchical network, 155
 - UDP, 152

- distributed region DDM, 169
 - distributed UDP forwarding, 149
 - federate for distributed forwarding, 146
 - FOM module, 191
 - full HLA compliance, 160
 - heartbeat interval, 91
 - HLA 1516 update requests, 131
 - implicit DDM, 171
 - join FOM module, 192
 - latency tests, 111
 - load-balanced forwarder group, 151
 - LRC response interval, 95
 - message compression, 127
 - MOM module, 192
 - multicast discovery, 124
 - multicast group, 133
 - network
 - buffer size, 129
 - congestion, 130
 - receive buffer, 129
 - packet bundling, 123
 - RID files, 62
 - routers and firewalls for distributed forwarding, 137
 - RTI Assistant, 67
 - messages, 69
 - RTI for FOM modules, 190
 - RTI Forwarder, 147
 - RTI Forwarders, 146
 - rtiexec, 66
 - save and restore, 97
 - sender-side filtering, 124, 126
 - silent failure, 96
 - single-portal forwarder group, 150
 - smart TCP forwarding, 126
 - switch values, 60
 - timeout
 - interval, 91
 - update rate reduction, 198
 - connecting
 - to RTI Forwarder, 94, 154
 - to rtiexec, 95
- 
- Connection command-line option, 73
 - consistency
 - RID file
 - checking, 62-63
 - RTI settings, 63
 - consistency checking
 - RTI settings, 223
 - console
 - clearing messages from, 72
 - pausing message updates, 72
 - saving to file, 72
 - constrained federate, 88
 - controlling
 - network congestion, 130
 - Convey Producing Federate
 - setting, 51
 - switch, 51
 - Convey Producing Federate switch, 220
 - Convey Region Designator
 - setting, 50
 - switch, 50
 - Convey Region Designator switch, 220
 - CPU, using, 106
 - CRC check, 225
 - create federation
 - FOM module, 190
 - create FOM module
 - configuring, 192
 - merging, 190
 - create module, 188
 - createFederationExecution, 39-40, 190
 - service, 86
 - createRTIambassador function, 61
 - creating a routes file, 147
 - current
 - FOM, 188
 - time, 88
- 
- d command-line option, 74, 76
- 
- D command-line option, 76
- 
- data
 - filtering update rate, 196
 - data distribution management, 106, See also DDM
 - service, 45, 209
 - DDM, 45, 66, 86, 106, 119, 124, 160, 174, 196, 209, 240
 - configuration parameters, 231
 - configuring
 - fixed grid, 169
 - data delay, 231
 - distributed region, 209
 - configuring, 169

- distributed region approach, 173
- example of, 237
- fixed grid, 209
- fixed grid DDM, 175
- implicit, 182
- interaction region information, 229
- region
 - implicit DDM, 184
- declaration management, 106, 132
 - multicast group, 133
- declaration management. See also DM, 106
- default
 - FED file, 224
- default time implementation, 225
- Defense Modeling and Simulation Office, 20
- delay
 - forwarding, 216
 - multicast discovery, 219
- Delay Subscription Evaluation
 - setting, 51
 - switch, 51
- Delay Subscription Evaluation switch, 220
- delay time
 - internal message, 163
- Delay Update Requests setting, 48
- delete command, 82
- deleting
 - federates and federation executions, 82
 - orphan, 47, 212
 - RTI settings, 58
- demilitarized zone, 137



- destAddrString command-line option, 76, 79
- destination address, 207
- destroyFederationExecution service, 86
- Detach Notify Level setting, 52
- diagnostic
 - configuring, 67
 - data, logging, 102
 - level, 74, 76, 78-79
 - messages, limiting, 67
- dimensions, implicit DDM, 184
- directory
 - log file, 52, 222
 - save restore, 46, 232
- Disable Unlicensed for Two setting, 57
- disabled service
 - warning, 53, 219
- disabled service exception, 53, 219



- disableMulticast command-line option, 74
- disabling
 - latency testing, 107
 - RID file consistency checking, 62
 - RTI Assistant, 73
 - save and restore, 45, 216
 - Shut Down All, 69
 - time management
 - display, 88
- discovered class advisory message, 210
- Discovery Message Delay setting, 47
- discovery processing
 - unknown objects, 96
- displaying
 - federate
 - information, 100
 - federation-wide messages, 90
 - FOM
 - interactions, 100-101
 - objects, 100
 - time management information, 88
- distributed forwarder
 - port, 214
- distributed forwarding, 30, 74
 - configuring
 - federate, 146
 - routers and firewalls, 155
 - forwarder group, 139
 - hierarchical network, 155
 - load-balanced, 144
 - optimizing on WAN, 143
 - routes file, 146
 - single-portal interconnection, 143
 - singleton network, 140
 - topologies
 - comparison, 141
 - LAN, 139
 - UDP, 152
 - configuring, 149
 - WAN topology, 142
- distributed region DDM, 173, 209
 - configuring, 169
 - distributed region matching, 174
 - modifying for implicit DDM, 185
- distributed singleton forwarder network, 140
- distributed UDP forwarding, 30
 - using unicast, 152



- distributedForwarderPort command-line option, 76
- distributing
 - FED file, 84
 - HLA 1516 DTD, 85
- DM, 106, 132, 196
 - multicast group, 133
- DM. See also declaration management, 106
- DMZ, 137
- Do not wait to send bundled packets setting, 56
- documentation
 - help, 13



- doNotUseRoutesFile command-line option, 76
- DTD, HLA 1516, 85
- dump, 52
- Dump RID setting, 53
- dumping
 - FED file, 223
 - RID file, 53, 223
- dynamic library, 21
- dynamic link compatibility, 23
- Dynamic Link Compatible 1516 API, 23
- dynamic linking, 21



- Enable Data Distribution Management setting, 45, 169
- Enable Exception Logging setting, 45, 161
- Enable Federate Heartbeat setting, 46, 91-92
- Enable Message Throttling RTI setting, 131
- Enable Message Throttling setting, 47, 132, 164-165
- Enable MOM setting, 45, 160-162
- Enable Name Reservation setting, 45
- Enable Packet Bundling setting, 56, 124
- Enable Reconnections setting, 46, 94-95
- Enable Save and Restore
 - RTI setting, 97
 - setting, 45
- Enable Send Internal RTI Messages Reliable (TCP)
 - setting, 84
- Enable TCP Compression setting, 56, 63, 128
- Enable Time Management setting, 45, 163
- Enable UDP Compression setting, 57, 63, 128

- Enable Update Rate Reduction
 - setting, 199
- Enable Update Rate Reduction setting, 45
- enabling
 - bundling, 148
 - compression, 148
 - latency testing, 107
 - RID file consistency checking, 62
 - save and restore, 45, 216
 - time management
 - display, 88
- environment variable
 - RTI_ASSISTANT_DISABLE, 73
 - RTI_ASSISTANT_IPV4_ADDRESS, 73
 - RTI_ASSISTANT_IPV6_ADDRESS, 73
 - RTI_ASSISTANT_PORT, 72
 - RTI_ASSISTANT_REMOVE_SHUT_DOWN_ALL, 69
 - RTI_ASSISTANT_TTL, 73
 - RTI_CONFIG, 43
 - RTI_FEDPRO_PORT, 31
 - RTI_RID_FILE, 43
- evokeCallback, 28
- example
 - building, 236
 - federates, 236
 - HLA Bounce, 240
 - rtisimple, 236
 - simpleDDM, 237
 - simpletime, 238
- exception
 - broken connection, 94
 - CallNotAllowedFromWithinCallback, 219
 - ConcurrentAccessAttempted, 219
 - disabled service, 53, 219
 - federate ambassador callback, 225
 - MOM, logging, 45, 232
 - throwing, 96
- Exception for Disable Service setting, 96
- Exception for Disabled Service setting, 53
- Exception Reporting
 - setting, 51, 161
 - switch, 51, 220
- executive, RTI, 66
- exiting
 - RTI Assistant, 69
- Extended HLA Version Compatibility setting, 50
- extension, FOM, 188
 - FOM module, using, 190



- F command-line option, 74, 76, 79



-f command-line option, 76, 79



factory, logical time, HLA 4, 40
 factory, logical time, HLA Evolved, 39
 failure, FOM module merge, 190
 fault tolerance, 86, 91
 orphaned objects, 92
 FDD file, 21, 84
 DTD, 85
 implicit DDM, 183
 FDD file. See also FED file, 21
 FED file, 22, 84, 86, 122
 compatibility, 21
 default, 224
 distributing, 84
 dumping, 223
 FOM module in, 190
 implicit DDM, 183
 search order, 84
 FED file. See also FDD file, 21
 federate
 abnormally terminated, 91
 configuring
 distributed forwarding, 146
 deleting, 82
 displaying information about, 100
 examples, 236
 filtering data, 196
 ID, 86
 interactions, 114
 latency test, 106
 listing, 82
 log, 101-102
 maximum number in federation, 49, 215
 network map, 88
 objects, 113
 protocol, 30
 command-line options, 78
 port, 31
 server, 77
 server, starting, 78
 usage, 31
 wrapper, 31
 regulating and constrained, 88
 RID file consistency, 60, 62
 statistic
 graphing, 116

federate ambassador callback
 exception, 225
 federate ID, 209
 Federate Reconnect Pause setting, 46, 94-95
 FederateAmbassador callback, 219
 federation
 data file, 21
 deleting, 82
 listing, 82
 maximum number of federates, 49, 215
 messages, 90
 name, 86
 save and restore, 66
 time, 38
 viewing details, 87
 federation execution. See federation, 21
 Federation Shadow, 93
 setting, 93



--federationHandle command-line option, 79



--federationName command-line option, 79



--Federations command-line option, 74

Federations View, 87
 connecting to RTI Forwarder, 154
 opening from command line, 74

file
 FED, 122
 MTL, 22, 204
 RID, 42, 122
 routes, 77
 RTI settings, 61

filtering
 implicit DDM, using, 182
 messages, 124, 126
 updates, 196

firewall
 configuring for distributed forwarding, 155

fixed grid
 configuration file, 209
 fixed grid DDM, 175, 209
 affect on RTI services, 180
 calculating axes, 177
 configuring, 169

- modifying for implicit DDM, 186
- range bounds, 180
- flushing, messages timeout, 225
- FOM
 - checking, 50, 225
 - current, 188
 - data, 136
 - displaying
 - interactions, 100-101
 - objects, 100
 - document data file, 21
 - extending using FOM module, 190
 - extension, 188
 - table, 191
- FOM Data Transport Type
 - setting, 122
- FOM Data Transport Type setting, 49
- FOM module, 233
 - class extension limits, 190
 - configuring, 191
 - failure mode, 190
 - in lightweight mode, 189
 - interoperability, 24
 - limitations on merging, 191
 - local, configuring, 193
 - merge order, 190
 - merging, 188, 232
 - module FOM, 188
 - MOM, create, and join, 188
 - RTI configuration for, 190
 - sorting, 232
 - specifying
 - create FOM module, 192
 - join FOM module, 192
 - table requirements, 190
 - validating, 190
- Force Unlicensed for Two setting, 57



- forceFullCompliance command-line option, 76



- forceShutdown command-line option, 74
- forceUnlicensedForTwo parameter, 218
- forwarder
 - address, 207
 - shutting down, 77
- forwarder group, 139, 214
 - compared to singleton forwarder, 141
 - configuring single-portal, 150

- Forwarder Listens to Local UDP Messages setting, 48
- Forwarder Log File Name setting, 52
- forwarder. See RTI Forwarder, 74



- forwarderPortToConnectTo command-line option, 76



- forwarderToConnectTo command-line option, 76
- forwarding
 - delay, 216
 - smart, 48-49, 215
- full compliance, 160, 206
- function
 - createRTIambassador, 61
 - tick(), 27-28

G

- graphing, federate statistics, 116
- group
 - forwarder, 214
 - multicast, 213



- H command-line option, 74



- h command-line option, 77

H

- handle
 - federate, 82
 - object instance
 - reusing, 210
- heartbeat, 46, 91, 212
 - interval, 46, 212
 - configuring, 91
 - setting, 91-92
 - timeout, 46, 212
- Heartbeat Interval setting, 46

help

HTML5, 13



--help command-line option, 77

hierarchical network, 155

configuring, 155

distributed forwarder, 155

High Level Architecture, See also HLA



--History command-line option, 74

HLA

API, optimizing performance, 106

compliance, configuring full, 160

example applications, 236

specification, 20

HLA 4

interoperability with HLA Evolved, 23

time implementation, 40

HLA 1516

DTD, 85

logical time, 39

HLA 1516 specifications

configuring, 131

specifying RID file, 61

HLA Bounce

example, 240

HLA Evolved

interoperability with HLA 4, 23

lightweight mode, 189

time implementation, 39

HLAfloat64Interval class, 40

HLAfloat64Time class, 40

HLAfloat64TimeFactory class, 40

HLAinteger64Interval class, 40

HLAinteger64Time class, 40

HLAinteger64TimeFactory class, 40

HLAlogicalTimeFactoryFactory

HLA Evolved, 39

HLAlogicalTimeFactoryFactory class, 40



-i command-line option, 80



I/O Period setting, 54

ID

federate, 209

IEEE 1516, 20, 22

API, 21



--ignore_rest command-line option, 74, 77-78, 80

implementing

logical time, 38

implicit DDM, 182

adding to FED or FDD file, 183

associating DDM regions, 184

configuring, 171

dimension, 184

initial bounds, 184

modifying distributed region extents, 185

modifying fixed grid DDM, 186

normalization, 184

importing

RTI settings, 58

interaction

displaying FOM, 100-101

federate, 114

sending, 122

Interaction Relevance Advisory

setting, 51

switch, 51

Interaction Relevance Advisory switch, 221

internal message, 136

configuring delay time, 163

reliable, 48, 208

enabling for FOM modules, 191

smart forwarding, 127

transmission rate, 164

interoperability

HLA 13 and 1516, 50, 209

HLA Evolved and HLA 4, 23

interval

heartbeat, 46, 212

response, 95

IO

count, 54, 230

queue, maximum, 54, 230

IP address, 76, 79, 123, 136, 207



Java

binding, 40

join FOM module

configuring, 192

- merging, 190
- join module, 188
- joinFederationExecution service, 86



- K command-line option, 73



- l command-line option, 77, 79



LAN

- configuring
 - best effort, 136
 - TCP forwarding, 136
- distributed forwarding
 - topologies, 139
 - topology comparisons, 141
- sending messages on, 29

latency, 28

- TCP, 55, 226

latency test, 106

- configuring, 111
- continuous, 111
- details, 110
- enabling or disabling, 107
- manual, 110
- results, 108
- running, 107

LBTS, 66, 88

libfedtime library, 34, 38

libfedtime1516 library, 34, 39

libfedtime1516_2025

- library, 40

libfedtime1516e

- library, 39

library

- for compiling, 34
- libfedtime, 38
- libfedtime1516, 39
- libfedtime1516_2025, 40
- libfedtime1516e, 39

librti1516, 34

libRTI-NG, 34

license, 57, 218

license checking, 57, 218

license checkout, 57, 218

lightweight mode, 85

- FOM modules, 189
- limitations, 86
- MOM DDM and time management, 66
- RTI Assistant, 85

link-compatibility, 21

linking

- dynamic, 21
- on Windows, 35-36, 38
- to 1.3, 35
- to 1516, 36

listing

- federations and federates, 82

load-balanced forwarder group

- configuring, 151

load-balanced forwarder network, 144

Local Component Notification History window, 90

local FOM module

- configuring, 193

Local RTI Component, 20, 42

log

- clearing, 72
- federate, 102
- file, 102
- LRC, 101
- pausing, 72
- rtiexec output, 67
- saving to file, 72
- window, 71

log file

- directory, 52, 222
- name, 52, 222
- notification level, 74, 76, 78-79
- reusing, 53, 222
- RTI Forwarder, 52, 222
- rtiexec, 52, 222
- rtiFederationShadow, 222

Log File Directory setting, 52

Log File Name

- setting, 102

log update interval

- RTI Assistant, 68

logical time

- 1516, 39
- implementing, 38

LogicalTime, 39-40

LogicalTime class, 40

LogicalTimeFactory class, 40

LogicalTimeFactoryFactory

- RTI 1516, 39

LogicalTimeFactoryFactory class, 39-40

LogicalTimeFactoryImpl class, 39

LogicalTimeFactoryImplFloat class, 39

- LogicalTimeFactoryImplInteger class, 39
- LogicalTimeImpl class, 39
- LogicalTimeImplFloat class, 39
- LogicalTimeImplInteger class, 39
- LogicalTimeInterval, 39-40
- LogicalTimeInterval class, 38, 40
- LogicalTimeIntervalImpl class, 39
- LogicalTimeIntervalImplFloat class, 39
- LogicalTimeIntervalImplInteger class, 39
- lookahead, 88
- lower bound time stamp, 66, 88
- LRC
 - delay in response to, 95
 - I/O thread, 28
 - log, 101
 - logging diagnostics, 102
 - message
 - federate, 102
 - reconnecting to RTI Forwarder, 94
- LRC Log File Name setting, 52
- LRC Minidumper Level setting, 52, 221



- M command-line option, 76, 79



- MAK Technologies Lisp, 22
- MÄK Technologies Lisp, 204
- manual
 - latency test, 110



- manual command-line option, 76, 79
- Max Forwarder Queue setting, 49
- Max I/O Count per Tick setting, 54
- Max I/O Queue setting, 54
- Max Number of Federates RTI setting, 124
- Max Number of Federates setting, 49
- Max Objects Per Federate setting, 49
- merge order
 - FOM module, 190
- merging
 - create FOM module, 190
 - FOM module, 232
 - FOM modules, 188, 190
 - join FOM module, 190
 - MOM module, 190

- message
 - asynchronous processing, 28
 - best effort, 136, 227
 - bundling, 47, 123, 216
 - compressing, 127
 - delay time for internal, 163
 - diagnostic, 67
 - federate
 - log, 102
 - federation-wide, 90
 - flushing, timeout, 225
 - forwarding internal, 127
 - internal
 - reliable, 48, 208
 - latency, 28
 - log
 - pausing updates, 72
 - RTI Assistant, 101
 - logging LRC, 101
 - notification, 52, 222
 - detaching from standard out, 52, 223
 - notification level, 74, 76, 78-79
 - queue, 49, 229
 - routing, 216
 - RTI Assistant
 - configuring, 69
 - sending
 - large, 128
 - throttling, 47, 228
 - transient
 - saving, 46, 217
 - transmit rate for internal, 164
- Messages page, 115
- minidump, 52
- Modeling and Simulation Coordination Office, 20
- modular FOM. See FOM module, 188
- module
 - FOM
 - in lightweight mode, 189
- modules
 - merging, 190
- MOM, 66, 86, 119, 160
 - exception logging, 45, 232
 - federate update interval, 45, 232
 - in FOM modules, 188
 - module, 188
 - service, 45, 208
- MOM Federate Update Interval setting, 45, 162
- MOM module
 - configuring, 192
 - merging, 190
 - specifying, 192
- MOM service reporting, 232

monitoring
network statistics, 55, 223

MTL, 22

file, 204

parameters, 205

multicast, 29-30, 138

address, 136

final, 231

RTI Assistant, 73

starting, 231

discovery, 218

address, 218

configuring, 124

delay, 219

port, 218

group, 132

address, 76

configuring, 133

TTL, 55, 226

multicast group, 213

address, 79

multicast TTL

RTI Assistant, 73

Multicast TTL setting, 55

multi-homed host, 133

multiple

address, 138

multithreaded application, 26

multithreading, 28



-N command-line option, 76, 80



name

log file, 52, 222

name reservation

in lightweight mode, 87

strict, 211

name reservation service, 45, 211

NAT, 155

network

compatibility, 22

congestion, 130

overhead, 106

statistics

monitoring, 55, 223

topologies

LAN, 29

network address translation, 155

network interface

address, 218

TCP, 218

network monitoring tool. See RTISpy Network

Monitoring window, 112

network receive buffer

configuring, 129

Network Statistics Monitoring

setting, 112

Network Statistics Monitoring setting, 55

network testing, 55, 223

Network Testing Enabled setting, 55



--networkInterfaceAddr command-line option, 76

next message request

service, 208

normalization

implicit DDM, 184

notification

configuring, 69

notification level

setting, 74, 76, 78-79

notify level, 52, 222

Notify Level setting, 52



object

attributes sent, 122

federate, 113

orphan, 91-92

unknown, 96

Object Class Relevance Advisory

setting, 51

switch, 51

Object Class Relevance Advisory switch, 221

object instance handle

reusing, 210

objects

displaying FOM, 100

maximum number per federate

federate

maximum number of objects, 215

optimizing

distributed forwarding on WAN, 143

performance, 106

orphan

deleting, 47, 212

federation shadow, 212

orphaned object, 91-92

Override FED Switches setting, 50
 Override FOM Switches, 161-162
 ownership management, 86, 240



-P command-line option, 77, 80



-p command-line option, 77-78, 80



packet

- best effort
 - address, 229
- bundling, 56, 123, 227
- forwarding
 - TCP, 29
- read per tick, 224
- UDP
 - size, 227

Packet Bundling Size setting, 56, 124

parameter

- commenting out, 204
- consistency of, 62
- forceUnlicensedForTwo, 218
- RID, 205-206, 217
- RTI_addForwarder, 148, 150, 156
- RTI_addForwarderConnection, 156
- RTI_addForwarderRoute, 156
- RTI_addressDelay, 169, 231
- RTI_-allowMulticastForwarding, 152
- RTI_allProvideUpdateRequestsDelayed, 229
- RTI_attributeRelevanceAdvisorySwitch, 221
- RTI_attributeScopeAdvisorySwitch, 221
- RTI_automaticNameReserveEnabled, 211
- RTI_autoProvideDelay, 229
- RTI_autoProvideSwitch, 220
- RTI_bestEffortSendRetries, 130, 227-228
- RTI_bestEffortSendRetryWaitUsec, 130, 227-228
- RTI_catchFedAmbExceptions, 225
- RTI_checkFlag, 224
- RTI_configureConnectionWithRid, 76, 103, 208, 218
- RTI_conveyOnlyAvailableDimensions, 231
- RTI_conveyProducingFederateSwitch, 220
- RTI_conveyRegionDesignatorSwitch, 220
- RTI_crcCheckFedFile, 225
- RTI_dataDistMgmt, 209
- RTI_ddmFixedGrid, 169, 209

- RTI_ddmFixedGridFilename, 169, 171, 209
- RTI_defaultFedFile, 84, 224
- RTI_defaultTimeImplementation, 225
- RTI_delaySubscriptionEvaluationSwitch, 220
- RTI_deleteOrphans, 212
- RTI_destAddrString, 76, 79, 103, 136, 138, 153, 207
- RTI_detachNotifyLevelFromStdOut, 223
- RTI_discoveryMsgBundlingDelay, 216
- RTI_distributedForwarderPort, 147-148, 155, 214
- RTI_distributedUdpForwarderMode, 138-139, 146, 152, 214
- RTI_distributeFedFile, 84, 190, 210, 225
- RTI_dualTransmitFirstInteractionRegions, 229
- RTI_dumpFed, 223
- RTI_dumpRid, 223
- RTI_enableBestEffortSendRetry, 130, 227-228
- RTI_enableFederateHeartbeat, 212
- RTI_enableFedexMsgRouting, 127, 216
- RTI_enableFomBackwardsCompatibility, 210
- RTI_enableHlaObjectNamePrefix, 210
- RTI_enableLRCNetworkStatisticsMonitoring, 223
- RTI_enableMessageThrottling, 228
- RTI_enableNameReservation, 211
- RTI_enableNetworkTesting, 223
- RTI_enablePacketBundling, 227
- RTI_enableSaveRestoreWhenUsingRtiexec, 216
- RTI_enableTcpCompression, 226
- RTI_enableUdpCompression, 226
- RTI_enableUpdateRateReduction, 217
- RTI_enableWarningsForDisabledServices, 219
- RTI_exceptionReportingSwitch, 220
- RTI_execMcastDiscoveryDelay, 219
- RTI_extend13and1516interop, 209
- RTI_federateHeartbeatInterval, 212
- RTI_federateReconnectPause, 94, 213
- RTI_federateTimeoutInterval, 212
- RTI_federationShadowEnabled, 212
- RTI_flushTimeoutInterval, 225
- RTI_fomDataTransportTypeControl, 103, 136, 208
- RTI_fomModuleMerging, 191, 232
- RTI_fomModuleSorting, 232
- RTI_forceFullCompliance, 103, 160, 206
- RTI_forceUnlicensedForTwo, 218
- RTI_forwarderGroupInterconnectMethod, 149-150, 156
- RTI_forwarderInterfaceAddr, 148
- RTI_forwarderRoutesFile, 77, 149, 214
- RTI_forwardingDelay, 216
- RTI_fullFedFileDistribution, 211
- RTI_implicitDdmDefaultSensorRange, 172
- RTI_implicitDdmDefaultUpdateRange, 172
- RTI_implicitDdmMaxExtentLimit, 172
- RTI_implicitDdmMinExtentLimit, 172

- RTI_implicitDdmOriginLat, 172
- RTI_implicitDdmOriginLong, 172
- RTI_implicitDdmParamsFile, 231
- RTI_implicitDdmRegionMargin, 172
- RTI_interactionRelevanceAdvisorySwitch, 221
- RTI_internalMsgReliableWhenUsingRtiexec, 84, 103, 136, 190, 208, 210
- RTI_internalMsgSmartForwardingWhenUsingRtiexec, 215
- RTI_IOPeriod, 230
- RTI_logFileDirectory, 222
- RTI_logfileName, 222
- RTI_lrcMcastDiscoveryDelay, 218
- RTI_lrcMinidumperLevel, 221
- RTI_maxChannelAddr, 169, 231
- RTI_maxForwarderQueue, 229
- RTI_maxIOCount, 230
- RTI_maxIOQueue, 230
- RTI_maxNumFederates, 215
- RTI_maxObjectsPerFederate, 215
- RTI_maxUdpPacketSize, 227
- RTI_mcastDiscoveryAddrString, 124, 218
- RTI_mcastDiscoveryEnabled, 103, 124, 218
- RTI_mcastDiscoveryPort, 218
- RTI_mcastDiscoveryTries, 124, 219
- RTI_mcastTtl, 226
- RTI_minChannelAddr, 169, 231
- RTI_momExceptionLogging, 232
- RTI_momFederateUpdateInterval, 232
- RTI_momModuleExtensionFileName, 191-192, 233
- RTI_momServiceAvailable, 208
- RTI_momVerboseLevel, 232
- RTI_networkInterfaceAddr, 218
- RTI_nextEventRequestAndRetractServiceEnabled, 208
- RTI_notifyLevel, 222
- RTI_objectClassRelevanceAdvisorySwitch, 221
- RTI_packetBundlingSize, 227
- RTI_preferLocalFomModule, 193, 233
- RTI_preferRidToFedSwitchTable, 220-221
- RTI_processingModel, 230
- RTI_processUnknownUpdatesForDiscovery, 96, 224
- RTI_receiveTolerance, 200, 217
- RTI_reconnectEnabled, 94, 213
- RTI_responseInterval, 209
- RTI_reuseLogFile, 222
- RTI_reuseReleasedObjectHandles, 210
- RTI_ridConsistencyChecking, 223
- RTI_routeToAllByDefault, 216
- RTI_RTI_disableUnlicensedForTwo, 218
- RTI_rtiExecLogFileName, 222
- RTI_rtiExecPerformsLicensing, 218
- RTI_rtiExecReconnectPause, 94, 213
- RTI_rtiForwarderLogFileName, 222
- RTI_saveRestoreDirectory, 232
- RTI_saveRestoreTimeout, 217
- RTI_saveTransientMessages, 217
- RTI_sendDiscoveredClass, 210
- RTI_serviceReportingSwitch, 220
- RTI_singleCallbackPerTick, 224
- RTI_smartForwardingLevel, 215
- RTI_socketReceiveBufferSize, 227
- RTI_socketSendBufferSize, 227
- RTI_strictFomChecking, 50, 225
- RTI
 - strictHLA1516eAttributeOwnershipUnavailableCallback, 211
- RTI_strictNameReservation, 211
- RTI_tcpBufferSize, 137, 226
- RTI_tcpCompressionLevel, 226
- RTI_tcpForwarderAddr, 91, 103, 124, 136-138, 146, 207
- RTI_tcpNetworkInterfaceAddr, 218
- RTI_tcpNoDelay, 226
- RTI_tcpPacketBundlingSize, 227
- RTI_tcpPort, 60, 77, 80, 103, 136, 207
- RTI
 - throwExceptionCallNotAllowedFromWithinCallback, 219
- RTI_throwExceptionForDisabledService, 219
- RTI_throwExceptionOnConcurrentAccess, 219
- RTI_tickFavorsNetwork, 230
- RTI_tickWaitPeriod, 230
- RTI_timeMgmt, 208
- RTI_transmitDelay, 228
- RTI_transmitRate, 228
- RTI_udpCompressionLevel, 226
- RTI_udpPacketBundlingSize, 227
- RTI_udpPort, 60, 77, 103, 136, 207
- RTI_use32BitsForValueSize, 128, 213
- RTI_useBusyWaitForTickMinMax, 228, 233
- RTI_useRandomNumberForFedHandle, 86, 209
- RTI_useRtiExec, 48, 60, 84, 103, 137, 190, 206, 208, 210
- RTI_waitIOPeriodForBundling, 227
- RTI-addCreateFomModule, 191-192, 233
- RTI-addDestAddrString, 138, 153, 229
- RTI-addDMInteractionMulticastAddr, 132-133, 214
- RTI-addDMObjectMulticastAddr, 132-133, 213
- RTI-addForwarderGroup, 149-150
- RTI-addForwarderGroupAddrString, 146, 214
- RTI-addForwarderToGroup, 149-150, 152
- RTI-addJoinFomModule, 192, 233
- RTI-addRidParametersToOverride, 63, 223
- RTI-addUpdateRate, 199-200, 217
- RTI-addUpdateRateSubscription, 200, 217
- RTI-allowMulticastForwarding, 149

- RTI-enableBundling, 148
- RTI-enableCompression, 148
- rtiFederationShadowLogFileName, 222
- RTI-removeRidParametersToOverride, 63, 223
- RTI-setObjectSensorRange, 172
- RTI-setObjectUpdateRange, 172
- RTI-setObjectWeaponRange, 172
- sending large, 128
- TimeToLive, 226
- passelization, 174
- pausing
 - message updates, 72
- performance, 26
 - affect of rtiexec use, 66
 - asynchronous I/O, 26
 - buffering update requests, 131
 - effect of tick() function, 27
 - improving with asynchronous options, 130
 - latency test, 106
 - message compression, 127
 - network congestion, 130
 - optimizing, 106
 - update rate reduction, 196
- period, polling, 54, 230
- plug-in
 - API, 22
- polling period, 54, 230
- pop-up message, configuring, 69
- port
 - distributed forwarder, 214
 - forwarder, 147
 - multicast discovery, 218
 - RTI Assistant, 72
 - setting for TCP, 77, 80
 - specifying, 77, 80
 - TCP, 207
 - UDP, 136, 207



- port command-line option, 78
- portal forwarder, 143-144
- primary forwarder, 139
- process model
 - RTI, 26
- processing
 - callbacks, 118
 - createFederation, destroyFederation, joinFederation, 95
 - unknown object, 224
- processing model, 54, 230

- Processing Model
 - setting, 131
- Processing Model setting, 54
- protocol, federate, 30
 - command-line options, 78
 - port, 31
 - server, 77
 - server, starting, 78
 - usage, 31
 - wrapper, 31
- Provide Delay
 - RTI setting, 131



- q command-line option, 67, 79



- queue
 - IO, maximum, 54, 230
 - message, 49, 229



- r command-line option, 77, 80



- R command-line option, 77, 79



- range
 - update, sensor, and weapon for implicit DDM, 182
- range bounds, 180
- receive buffer, 129
 - message queue, 49, 229
- receive tolerance, 217
- reconnect
 - TCP, 46, 213
- reconnecting
 - RTI Forwarder, 46, 213
 - to RTI Forwarder, 94
- region
 - associating with implicit DDM, 184
- region matching, 174
- registering
 - update rate, 217
- regulating federate, 88

- reliable transport, 29, 66, 86, See also TCP
 - choosing, 122
 - internal, 48, 208
 - topologies, 29
 - WAN, 30
- remote assistant
 - connecting to, 69
- reporting, MOM service, 232
- response interval, 48, 95, 209
- Response Interval
 - setting, 95-96
- Response Interval setting, 48
- restore
 - configuring, 97
 - directory, 46, 232
 - enabling and disabling, 45, 216
 - timeout, 45, 217
- retract service, 208
- Reuse Log File setting, 53
- reusing
 - log file, 53, 222
 - object instance handle, 210
- RID file, 42, 122, 205
 - adding, 57
 - compatibility, 22
 - consistency, 62
 - among federates, 60
 - displaying settings, 59
 - dumping, 53, 223
 - FOM module parameters, 191
 - HLA 1516 specifications
 - specifying programmatically, 61
 - parameters, 206, 217
 - search order, 42
- rid.mtl, 22, 42, 205
- router
 - configuring for distributed forwarding, 155
- routes file, 77, 214, See also routes.mtl
 - creating, 147
 - RTI forwarder, configure, 146
 - search order, 149
- routes.mtl, 146, See also routes file
 - for firewalls and routers, 155
 - search order, 149
- Initialization Data file, 42
- message types, 115
- process model, 26
- specification, 20
- RTI 1516
 - LogicalTimeFactoryFactory, 39
 - name reservation, 87
 - supported services, 22
- RTI Ambassador, 115
- RTI Assistant
 - adding RTI settings configurations, 57
 - address, 73
 - command-line options, 73
 - configuring, 67
 - connecting to remote, 69
 - disabling, 73
 - displaying messages, 90
 - lightweight mode, 85
 - log update interval, 68
 - logging messages, 101
 - multicast TTL, 73
 - pop-ups and notifications, 69
 - port, 72
 - shutting down, 69
- RTI Chooser, 43
- RTI Exec Log File Name setting, 52
- RTI Exec Performs Licensing setting, 57
- RTI Federate Protocol Server
 - command-line options, 78
- RTI Federation Shadow
 - command-line options, 79
- RTI Forwarder, 30, 74, 91, 124
 - command-line options, 76
 - configuring, 146-147
 - hierarchical network, 155
 - load-balanced group, 151
 - configuring for routers and firewalls, 137
 - configuring single-portal, 150
 - connection to, 154
 - group, 139
 - hierarchical network
 - topology, 155
 - load-balanced forwarder network, 144
 - multiple on LAN, 139
 - portal, 143-144
 - primary and ancillary, 139
 - reconnecting, 46, 213
 - reconnecting to, 94
 - routes file, 77
 - shutting down, 77
 - singleton, 140
 - UDP, 152
 - using unicast for UDP, 152



--routesFileName command-line option, 77

RTI

- compatibility, 20
- configuring for FOM modules, 190
- definition, 20, 22
- executive, 66

- RTI Forwarder log file, 52, 222
- RTI initialization data file, 22
- RTI services
 - affect of fixed grid DDM, 180
- RTI setting
 - All Provides Delayed, 131
 - Enable Message Throttling, 131
 - Enable Save and Restore, 97
 - Max Number of Federates, 124
 - Provide Delay, 131
 - Save/Restore Timeout, 97
- RTI settings
 - adding configurations in RTI Assistant, 57
 - checking consistency, 223
 - consistency, 63
 - deleting, 58
 - file, 61
 - importing, 58
- RTI_addForwarder parameter, 148, 150, 156
- RTI_addForwarderConnection parameter, 156
- RTI_addForwarderRoute parameter, 156
- RTI_addressDelay parameter, 169, 231
- RTI_allProvideUpdateRequestsDelayed parameter, 229
- RTI_ASSISTANT_DISABLE
 - environment variable, 73
- RTI_ASSISTANT_IPV4_ADDRESS
 - environment variable, 73
- RTI_ASSISTANT_IPV6_ADDRESS
 - environment variable, 73
- RTI_ASSISTANT_PORT environment variable, 72
- RTI_ASSISTANT_REMOVE_SHUT_DOWN_ALL
 - environment variable, 69
- RTI_ASSISTANT_TTL
 - environment variable, 73
- RTI_attributeRelevanceAdvisorySwitch
 - parameter, 221
- RTI_attributeScopeAdvisorySwitch parameter, 221
- RTI_automaticNameReserveEnabled parameter, 211
- RTI_autoProvideDelay parameter, 229
- RTI_autoProvideSwitch parameter, 220
- RTI_bestEffortSendRetries parameter, 130, 227-228
- RTI_bestEffortSendRetryWaitUsec parameter, 130, 227-228
- RTI_catchFedAmbExceptions parameter, 225
- RTI_checkFlag parameter, 224
- RTI_CONFIG environment variable, 43, 84
- RTI_configureConnectionWithRid parameter, 76, 103, 208, 218
- RTI_conveyOnlyAvailableDimensions parameter, 231
- RTI_conveyProducingFederateSwitch parameter, 220
- RTI_conveyRegionDesignatorSwitch parameter, 220
- RTI_crcCheckFedFile parameter, 225
- RTI_dataDistMgmt parameter, 209
- RTI_ddmFixedGrid parameter, 169, 209
- RTI_ddmFixedGridFilename parameter, 169, 171, 209
- RTI_defaultFedFile parameter, 84, 224
- RTI_defaultTimeImplementation parameter, 225
- RTI_delaySubscriptionEvaluationSwitch
 - parameter, 220
- RTI_deleteOrphans parameter, 212
- RTI_destAddrString parameter, 76, 79, 103, 136, 138, 153, 207
- RTI_detachNotifyLevelFromStdOut parameter, 223
- RTI_discoveryMsgBundlingDelay parameter, 216
- RTI_distributedForwarderPort parameter, 147-148, 155, 214
- RTI_distributedUdpForwarderMode parameter, 138-139, 146, 152, 214
- RTI_distributeFedFile parameter, 84, 190, 210, 225
- RTI_dualTransmitFirstInteractionRegions
 - parameter, 229
- RTI_dumpFed parameter, 223
- RTI_dumpRidparameter, 223
- RTI_enableBestEffortSendRetry parameter, 130, 227-228
- RTI_enableFederateHeartbeat parameter, 212
- RTI_enableFedexMsgRouting parameter, 127, 216
- RTI_enableFomBackwardsCompatibility
 - parameter, 210
- RTI_enableHlaObjectNamePrefix parameter, 210
- RTI_enableLRCNetworkStatisticsMonitoring
 - parameter, 223
- RTI_enableMessageThrottling parameter, 228
- RTI_enableNameReservation parameter, 211
- RTI_enableNetworkTesting parameter, 223
- RTI_enablePacketBundling parameter, 227
- RTI_enableSaveRestoreWhenUsingRtiexec
 - parameter, 216
- RTI_enableTcpCompression parameter, 226
- RTI_enableUdpCompression parameter, 226
- RTI_enableUpdateRateReduction parameter, 217
- RTI_enableWarningsForDisabledServices
 - parameter, 219
- RTI_exceptionReportingSwitch parameter, 220
- RTI_execMcastDiscoveryDelay parameter, 219

- RTI_extend13and1516interop parameter, 209
- RTI_federateHeartbeatInterval parameter, 212
- RTI_federateReconnectPause parameter, 94, 213
- RTI_federateTimeoutInterval parameter, 212
- RTI_federationShadowEnabled parameter, 212
- RTI_FEDPRO_PORT environment variable, 31
- RTI_flushTimeoutInterval parameter, 225
- RTI_fomDataTransportTypeControl parameter, 103, 136, 208
- RTI_fomModuleMerging parameter, 191, 232
- RTI_fomModuleSorting parameter, 232
- RTI_forceFullCompliance parameter, 103, 160, 206
- RTI_forceUnlicensedForTwo parameter, 218
- RTI_forwarderGroupInterconnectMethod parameter, 149-150, 156
- RTI_forwarderInterfaceAddr parameter, 148
- RTI_forwarderRoutesFile parameter, 77, 149, 214
- RTI_forwardingDelay parameter, 216
- RTI_fullFedFileDistribution parameter, 211
- RTI_implicitDdmDefaultSensorRange parameter, 172
- RTI_implicitDdmDefaultUpdateRange parameter, 172
- RTI_implicitDdmMaxExtentLimit parameter, 172
- RTI_implicitDdmMinExtentLimit parameter, 172
- RTI_implicitDdmOriginLat parameter, 172
- RTI_implicitDdmOriginLong parameter, 172
- RTI_implicitDdmParamsFile parameter, 231
- RTI_implicitDdmRegionMargin parameter, 172
- RTI_interactionRelevanceAdvisorySwitch parameter, 221
- RTI_internalMsgReliableWhenUsingRtiexec parameter, 84, 103, 136, 190, 208, 210
- RTI_ internalMsgSmartForwardingWhenUsingRtiexec parameter, 215
- RTI_IOPeriod parameter, 230
- RTI_logFileDirectory parameter, 222
- RTI_logfileName parameter, 222
- RTI_lrcMcastDiscoveryDelay parameter, 218
- RTI_lrcMinidumperLevel parameter, 221
- RTI_maxChannelAddr parameter, 169, 231
- RTI_maxForwarderQueue parameter, 229
- RTI_maxIOCount parameter, 230
- RTI_maxIOQueue parameter, 230
- RTI_maxNumFederates parameter, 215
- RTI_maxObjectsPerFederate parameter, 215
- RTI_maxUdpPacketSize parameter, 227
- RTI_mcastDiscoveryAddrString parameter, 124, 218
- RTI_mcastDiscoveryEnabled parameter, 103, 124, 218
- RTI_mcastDiscoveryPort parameter, 218
- RTI_mcastDiscoveryTries parameter, 124, 219
- RTI_mcastTtl parameter, 226
- RTI_minChannelAddr parameter, 169, 231
- RTI_momExceptionLogging parameter, 232
- RTI_momFederateUpdateInterval parameter, 232
- RTI_momModuleExtensionFileName parameter, 191-192, 233
- RTI_momServiceAvailable parameter, 208
- RTI_momVerboseLevel parameter, 232
- RTI_networkInterfaceAddr parameter, 218
- RTI_nextEventRequestAndRetractServiceEnabled parameter, 208
- RTI_notifyLevel parameter, 222
- RTI_objectClassRelevanceAdvisorySwitch parameter, 221
- RTI_packetBundlingSize parameter, 227
- RTI_preferLocalFomModule parameter, 193, 233
- RTI_preferRidToFedSwitchTable parameter, 220-221
- RTI_processingModel parameter, 230
- RTI_processUnknownUpdatesForDiscovery parameter, 96, 224
- RTI_receiveTolerance parameter, 200, 217
- RTI_reconnectEnabled parameter, 94, 213
- RTI_responseInterval parameter, 209
- RTI_reuseLogFile parameter, 222
- RTI_reuseReleasedObjectHandles parameter, 210
- RTI_RID_FILE environment variable, 43 setting programmatically, 61
- RTI_ridConsistencyChecking parameter, 223
- RTI_routeToAllByDefault parameter, 216
- RTI_RTI_disableUnlicensedForTwo parameter, 218
- RTI_rtiExecLogFileName parameter, 222
- RTI_rtiExecPerformsLicensing parameter, 218
- RTI_rtiExecReconnectPause parameter, 94, 213
- RTI_rtiForwarderLogFileName parameter, 222
- RTI_saveRestoreDirectory parameter, 232
- RTI_saveRestoreTimeout parameter, 217
- RTI_saveTransientMessages parameter, 217
- RTI_sendDiscoveredClass parameter, 210
- RTI_serviceReportingSwitch parameter, 220
- RTI_singleCallbackPerTick parameter, 224
- RTI_smartForwardingLevel parameter, 215
- RTI_socketReceiveBufferSize parameter, 227
- RTI_socketSendBufferSize parameter, 227
- RTI_strictFomChecking parameter, 50, 225
- RTI_ strictHLA1516eAttributeOwnershipUnavailableCallback parameter, 211
- RTI_strictNameReservation parameter, 211
- RTI_tcpBufferSize parameter, 137, 226
- RTI_tcpCompressionLevel parameter, 226

- RTI_tcpForwarderAddr parameter, 91, 103, 124, 136-138, 146, 207
- RTI_tcpNetworkInterfaceAddr parameter, 218
- RTI_tcpNoDelay parameter, 226
- RTI_tcpPacketBundlingSize parameter, 227
- RTI_tcpPort parameter, 60, 77, 80, 103, 136, 207
- RTI_
 - throwExceptionCallNotAllowedFromWithinCallback parameter, 219
- RTI_throwExceptionForDisabledService parameter, 219
- RTI_throwExceptionOnConcurrentAccess parameter, 219
- RTI_tickFavorsNetwork parameter, 230
- RTI_tickWaitPeriod parameter, 230
- RTI_timeMgmt parameter, 208
- RTI_transmitDelay parameter, 228
- RTI_transmitRate parameter, 228
- RTI_udpCompressionLevel parameter, 226
- RTI_udpPacketBundlingSize parameter, 227
- RTI_udpPort parameter, 60, 77, 103, 136, 207
- RTI_use32BitsForValueSize parameter, 128, 213
- RTI_useBusyWaitForTickMinMax parameter, 228, 233
- RTI_useRandomNumberForFedHandle parameter, 86, 209
- RTI_useRtiExec parameter, 48, 60, 84, 103, 137, 190, 206, 208, 210
- RTI_waitIOPeriodForBundling parameter, 227
- RTI-addCreateFomModule parameter, 191-192, 233
- RTI-addDestAddrString parameter, 138, 153, 229
- RTI-addDMInteractionMulticastAddr parameter, 132-133, 214
- RTI-addDMObjectMulticastAddr parameter, 132-133, 213
- RTI-addForwarderGroup parameter, 149-150
- RTI-addForwarderGroupAddrString parameter, 146, 214
- RTI-addForwarderToGroup parameter, 149-150, 152
- RTI-addJoinFomModule parameter, 192, 233
- RTI-addRidParametersToOverride parameter, 63, 223
- RTI-addUpdateRate parameter, 199-200, 217
- RTI-addUpdateRateSubscription parameter, 200, 217
- RTI-allowMulticastForwarding parameter, 149, 152
- RTIambassador service, 219
- RTI-enableBundling parameter, 148
- RTI-enableCompression parameter, 148
- rtiexec, 66
 - configuring, 66
 - consistency of RID parameters, 62
 - enabling for FOM modules, 191
 - latency testing, 107
 - log file, 52, 222
 - logging
 - diagnostics, 102
 - output, 67
 - message routing, 216
 - required use, 66
 - response time, 95
 - running RTI without, 85
 - save and restore, 45, 216
 - specifying FED file, 84
 - synchronization points, 89
 - time management display, 88
 - using, 206
 - viewing federation details, 87
- Rtiexec Reconnect Pause setting, 47
- rtiFederationShadow
 - log file, 222
- rtiFederationShadowLogFileName parameter, 222
- RTIfedTime class, 38
- RTIinternalError, 94
- RTI-removeRidParametersToOverride parameter, 63, 223
- RTI-setObjectSensorRange parameter, 172
- RTI-setObjectUpdateRange parameter, 172
- RTI-setObjectWeaponRange parameter, 172
- rtisimple example, 236
- RTIspy
 - API, 22
 - displaying
 - FOM interactions, 100-101
 - FOM objects, 100
- RTIspy Network Monitoring window
 - Messages page, 115
- running latency test, 107
- Run-Time Infrastructure, 22

S

save

- configuring, 97
- directory, 46, 232
- enabling and disabling, 45, 216
- timeout, 45, 217

Save and Restore, 86

Save Restore Directory setting, 46

Save Restore Timeout setting, 45

Save Transient Messages setting, 46

Save/Restore Timeout RTI setting, 97

saving

- messages to file, 72
- transient messages, 46, 217

scaffolding definition, 190

search order

- FED file, 84
- fixed grid configuration file, 171
- RID file, 42
- routes file, 149

send buffer, 129

Send Internal Messages Reliable (TCP) setting, 122

Send Internal RTI Messages Reliable (TCP) setting, 48

sender-side filtering, 124, 126

- internal messages, 127

sending

- advisory messages, 60
- large attributes and parameters, 128
- messages
 - LAN, 29
 - WAN, 30
- object attributes and interactions, 122

sensor range, 182

server, RTI central, 66

service

- createFederation, 95
- createFederationExecution, 86
- data distribution management, 45, 209
- destroyFederation, 95
- joinFederation, 95
- joinFederationExecution, 86
- MOM, 45, 208
- name reservation, 45, 211
- next message request, 208
- retract, 208
- RTIambassador, 219
- supported for 1516, 22

Service Reporting

- setting, 51, 162

switch, 51, 220

-

--setLogFileName command-line option, 77, 79

-

--setNotifyQuiet command-line option, 79

-

--setRidFileName command-line option, 77, 79

-

--setTcpPort command-line option, 77, 80

setting

- Attribute Relevance Advisory, 51
- Attribute Scope Advisory, 51
- Auto Provide Delay, 47
- Automatically Delete Orphans, 47, 92-94
- Auto-Provide, 50
- Convey Producing Federate, 51
- Convey Region Designator, 50
- Delay Subscription Evaluation, 51
- Delay Update Requests, 48
- Detach Notify Level, 52
- Disable Unlicensed for Two, 57
- Discovery Message Delay, 47
- Do not wait to send bundled packets, 56
- Dump RID, 53
- Enable Data Distribution Management, 45, 169
- Enable Exception Logging, 45, 161
- Enable Federate Heartbeat, 46, 91-92
- Enable Message Throttling, 47, 132, 164-165
- Enable MOM, 45, 160-162
- Enable Name Reservation, 45
- Enable Packet Bundling, 56, 124
- Enable Reconnections, 46, 94-95
- Enable Save and Restore, 45
- Enable Send Internal RTI Messages Reliable (TCP), 84
- Enable TCP Compression, 56, 63, 128
- Enable Time Management, 45, 163
- Enable UDP Compression, 57, 63, 128
- Enable Update Rate Reduction, 45, 199
- Exception for Disable Service, 96
- Exception for Disabled Service, 53
- Exception Reporting, 51, 161
- Extended HLA Version Compatibility, 50
- Federate Reconnect Pause, 46, 94-95

- FOM Data Transport Type, 49, 122
- Force Unlicensed for Two, 57
- Forwarder Listens to Local UDP Messages, 48
- Forwarder Log File Name, 52
- Heartbeat Interval, 46
- heartbeat interval, 91-92
- I/O Period, 54
- Interaction Relevance Advisory, 51
- Log File Directory, 52
- Log File Name, 102
- LRC Log File Name, 52
- LRC Minidumper Level, 52
- Max Forwarder Queue, 49
- Max I/O Count per Tick, 54
- Max I/O Queue, 54
- Max Number of Federates, 49
- Max Objects Per Federate, 49
- MOM Federate Update Interval, 45, 162
- Multicast TTL, 55
- Network Statistics Monitoring, 55, 112
- Network Testing Enabled, 55
- Notify Level, 52
- Object Class Relevance Advisory, 51
- Override FED Switches, 50
- Packet Bundling Size, 56, 124
- Processing Model, 54, 131
- Response Interval, 48, 95-96
- Reuse Log File, 53
- RTI Exec Log File Name, 52
- RTI Exec Performs Licensing, 57
- RTI, adding, 57
- Rtiexec Reconnect Pause, 47
- Save Restore Directory, 46
- Save Restore Timeout, 45
- Save Transient Messages, 46
- Send Internal Messages Reliable (TCP), 122
- Send Internal RTI Messages Reliable (TCP), 48
- Service Reporting, 51, 162
- Smart Forwarding Level, 49, 126-127
- Socket Receive Buffer Size, 55, 130
- Socket Send Buffer Size, 56, 130
- Strict Attribute Ownership Unavailable, 50
- switch values, 60
- TCP Buffer Size, 55, 130, 137
- TCP Compression Level, 56, 63
- TCP No Delay, 55
- Tick Wait Period, 54
- Timeout Interval, 46, 92
- timeout interval, 91
- Transmit Delay, 47, 164
- Transmit Rate, 47, 165
- UDP Compression Level, 57, 63
- Use Smart Forwarding with Internal Messages, 48
- Warnings for Disabled Service, 53
- 
- setUdpPort command-line option, 77, 80
- Shut Down All
 - disabling, 69
- 
- shutdownRequest command-line option, 74
- shutting down
 - RTI Assistant, 69
 - RTI Forwarder, 77
- silent failure, 96
- simpleDDM example, 237
- simpletime example, 238
- simulation time, 88
- single-portal forwarder group, configuring, 150
- single-portal interconnection, 143
- singleton forwarder, 140
 - compared to forwarder group, 141
- SISO DLC HLA API 1516, 21
- SISO Dynamic Link Compatibility HLA API
 - Product Development Group, 23
- SISO-STD-004.1-2004, 21-22
- size, bundle, 148
- smart forwarding, 48-49, 215
- Smart Forwarding Level setting, 49, 126-127
- smart TCP forwarding, 124, 126, See also sender-side filtering
- socket
 - receive buffer, 55, 227
 - send buffer, 56, 227
 - TCP_NODELAY, 55, 226
- Socket Receive Buffer Size setting, 55, 130
- Socket Send Buffer Size setting, 56, 130
- sorting
 - FOM module, 232
- specification
 - obtaining HLA and RTI, 20
- specifying
 - compliant mode, 76
 - notification level, 74, 76, 78-79
 - port, 77, 80
 - Warning Tick Interval, 71
- standard out
 - detaching messages from, 52, 223
- static routing, 137
- statistics
 - graphing, 116

- network
 - monitoring, 55, 223
- Strict Attribute Ownership Unavailable setting, 50
- subscription, 240
- switch
 - Attribute Relevance Advisory, 51, 221
 - Attribute Scope Advisory, 51, 221
 - Convey Producing Federate, 51, 220
 - Convey Region Designator, 50, 220
 - Delay Subscription Evaluation, 51, 220
 - Exception Reporting, 51, 220
 - Interaction Relevance Advisory, 51, 221
 - Object Class Relevance Advisory, 51, 221
 - Service Reporting, 51, 220
- switch table, 50, 220
- switch values, setting, 60
- synchronization point, 66, 86
 - viewing, 89
- synchronous mode, 54, 230
- synchronous processing, 27
-
- T command-line option, 77, 80
-
- t command-line option, 77, 80
- T
- table
 - FOM, 191
 - FOM in FOM modules, 190
- TCP, 66, 122-123, See also reliable transport
 - buffer, 55, 129, 226
 - compressing messages, 127
 - compression, 56, 63, 226
 - latency, 55, 226
 - network interface address, 218
 - packet, 123
 - port, 207
 - reconnect, 46, 213
 - setting port, 77, 80
 - throughput, 55, 226
 - topologies, 29
- TCP Buffer Size setting, 55, 130, 137
- TCP Compression Level setting, 56, 63
- TCP Forwarder address, 207
- TCP forwarding, 29-30
 - configuring centralized, 136
 - configuring smart, 126
 - internal messages, 127
 - on WAN, 137
 - smart, 124
- TCP No Delay setting, 55
- TCP_NODELAY socket, 55, 226
-
- tcpAddress command-line option, 77, 80
-
- tcpNetworkInterfaceAddr command-line option, 80
- terminated federate, 91
- testing
 - latency, 106
 - network, 55, 223
- thread
 - asynchronous I/O, 28
 - federate and asynchronous, 28
- thread-safety, 26, 130
- throttling message, 47, 228
- throughput, TCP, 55, 226
- tick, 228, 230
 - packets read per, 224
 - wait period, 54, 230
- Tick Wait Period setting, 54
- tick() function, 26-28
 - not calling, 29
- tick(min,max), 233
- time
 - advance grant, 88
 - federation, 38
 - HLA 4
 - logical time, 40
 - HLA Evolved
 - logical time, 39
 - simulation, 88
- time implementation
 - default, 225
 - libfedtime, 38
- time management, 40, 45, 66, 86, 160, 208
 - display, 88
 - example, 238
 - time implementation, 38
- timeout
 - heartbeat, 46, 212
 - interval
 - configuring, 91
 - setting, 91-92

- message flushing, 225
- save restore, 45, 217
- Timeout Interval setting, 46
- TimeToLive parameter, 226
- topology
 - distributed forwarding
 - LAN, 139
 - WAN, 142
 - network, 29
- transient message
 - saving, 217
- transient message, saving, 46
- Transmit Delay setting, 47, 164
- transmit rate, 164
- Transmit Rate setting, 47, 165
- TTL, multicast, 55, 226

U

- UDP, 122-123, See also best effort
 - compressing messages, 127
 - compression, 57, 63, 226
 - datagram, 123
 - distributed forwarding, 152
 - forwarding, 30
 - using unicast, 152
 - over WAN, 138
 - packet size, 227
 - port, 207
 - setting for best effort, 136
 - topologies, 29
- UDP Compression Level setting, 57, 63
- UDP forwarder, 214

-

- udpNetworkInterfaceAddr command-line option, 80
- unicast, distributed UDP forwarding, 152
- unimplemented service, responding to, 96
- UNIX, linking on, 35-37
- unknown objects
 - discovery processing for, 96
 - processing, 224
- unsubscription, 240
- update
 - filtering, 196
 - range, 182
 - reducing rate of, 196
 - request
 - buffering, 131

- update rate
 - reduction, 45, 196, 217
 - configuring, 198
 - registering, 217
- Use Smart Forwarding with Internal Messages
 - setting, 48

-

- usePreconfiguredRid command-line option, 77, 80

-

- useReliable command-line option, 80

-

- v command-line option, 77

V

- validating FOM module, 190
- validity attribute, 224
- version
 - displaying, 74, 78, 80

-

- version command-line option, 77
- viewing
 - federation details, 87
 - synchronization points, 89
 - time management information, 88
- virtual LAN, 137
- virtual private network, 137
- VPN, 137

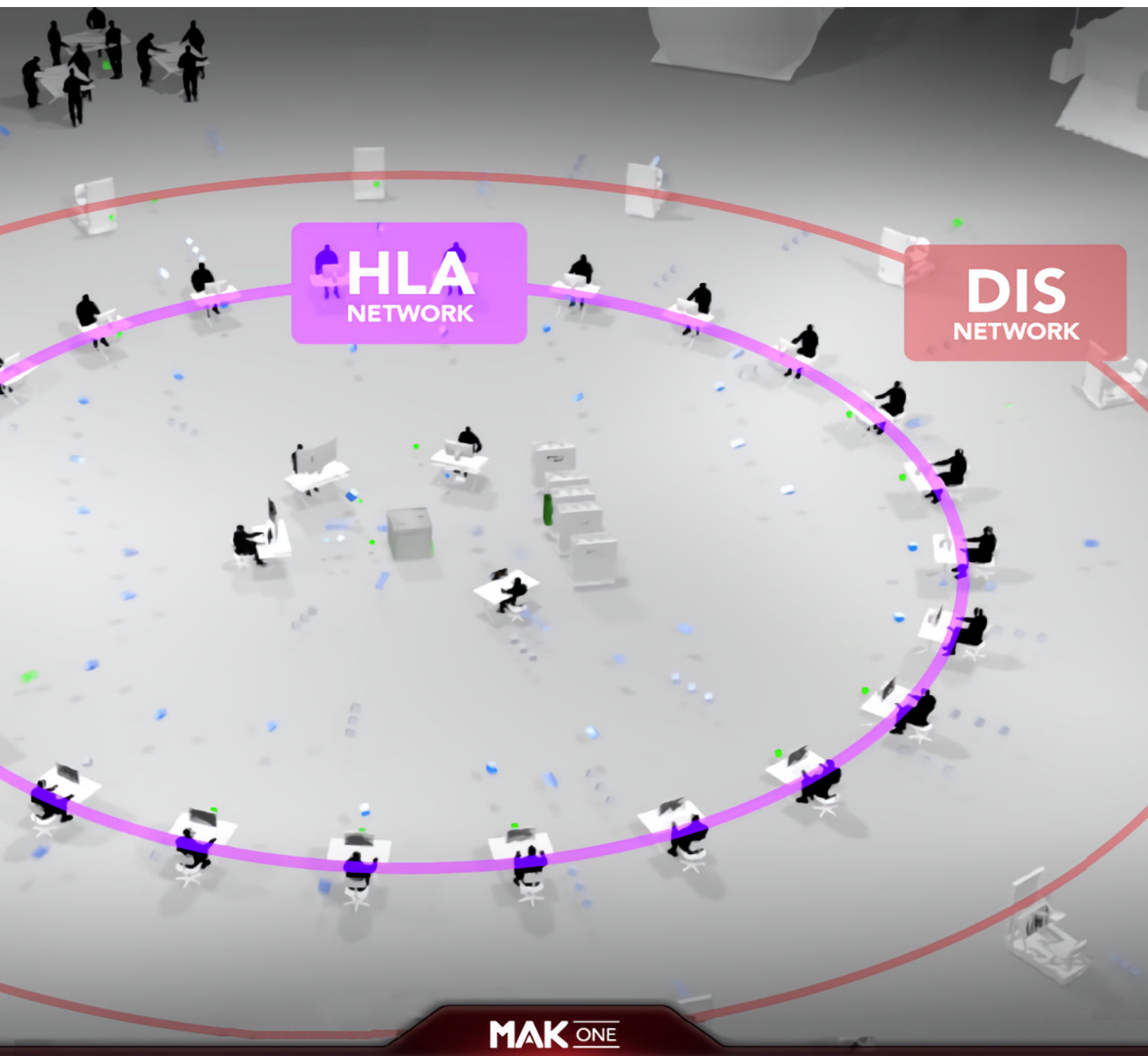
W

- wait period, tick, 54, 230
- WAN
 - centralized TCP forwarding, 137
 - communicating over, 138
 - configuring transport type, 137
 - connecting to assistants over, 69
 - distributed forwarding
 - optimizing, 143
 - single-portal interconnection, 143

- topology, 142
- load-balanced forwarder network, 144
- sending messages on, 30
- warning for disabled service, 53, 219
- Warning Tick Interval, 71
- Warnings for Disabled Service setting, 53
- weapon range, 182
- window, console, log, 71
- Windows
 - compiling and linking on, 35-36, 38
- wrapper
 - API, 32
 - C++, 31
 - federate ambassador, 31
 - service callbacks, 32
 - service calls, 31



XML file, 21



RTI-5.0-08-250702