



MÄK Data Logger

Users Guide



VT MÄK
A company of VT Systems



MÄK Data Logger

Users Guide

Copyright © 2016 VT MÄK
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 VT MÄK.

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

All other trademarks are owned by their respective companies.

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

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

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

info@mak.com
www.mak.com

Revision LOG-5.4-1-160325



Contents

Preface

How the Manual is Organized	xi
Logger Documentation	xii
MAK Products	xiii
How to Contact Us	xv
Document Conventions	xvi
Mouse Button Naming Conventions.....	xvii
Third Party Licenses	xvii
Boost License.....	xvii
libXML and libICONV	xviii

Chapter 1. Introduction

1.1. Logger Features	1-2
1.1.1. Logger Interfaces	1-3
1.1.2. Filtering	1-3
1.1.3. Annotations	1-3
1.1.4. Editing	1-4
1.1.5. Data Export	1-4
1.1.6. HLA Traffic Analyzer	1-4
1.1.7. The Remote Control Logger	1-4
1.1.8. Sample Logger Files	1-5
1.1.9. Sample Programs	1-6
1.2. Distributed Simulation Standards Supported	1-7
1.2.1. RTI Support	1-7
1.2.2. FOM Support	1-7

Chapter 2. Installing the Logger

2.1. Installing the MAK Data Logger	2-2
2.1.1. Installing the MAK Data Logger on Windows	2-2

2.1.2. Installing the Logger on a Linux System	2-2
2.2. The Logger Directory Structure	2-3
2.3. Installing and Setting Up the MÄK License Manager	2-4
2.3.1. Specifying the License Server	2-5
2.4. Installing an RTI	2-7
2.4.1. Installing the MÄK RTI	2-8
2.5. Localizing the Graphical User Interface	2-9
2.5.1. Translating Other Interface Files	2-12
2.5.2. Applying the Language Files	2-13

Chapter 3. Logger Concepts

3.1. What the Logger Does During Recording and Playback	3-2
3.1.1. The Exercise Connection	3-2
3.1.2. What The Logger Does During Recording	3-3
3.1.3. What The Logger Does During Playback	3-3
3.1.4. Recording and Playback Using HLA Time Management	3-4
3.2. Timescale During Playback	3-6
3.2.1. Using Time Scaling with Time Management	3-6
3.2.2. Compensating for Timescale Effects During Playback	3-6
3.3. HLA Support	3-8
3.3.1. FOM Flexibility	3-8
3.3.2. Backward Compatibility	3-8

Chapter 4. Configuring the Logger Connection

4.1. Configuring the Exercise Connection	4-3
4.1.1. Viewing Connection Properties	4-3
4.1.2. Editing a Connection	4-4
4.2. Configuring the Logger for HLA	4-6
4.2.1. Specifying the Name of the Logger Federate (HLA Evolved Only)	4-7
4.2.2. Specifying the Type of the Logger Federate	4-7
4.2.3. Specifying the Federation Execution Name	4-8
4.2.4. Specifying a FED File	4-8
4.2.5. Specifying FOM Modules (HLA Evolved Only)	4-9
4.2.6. Specifying the MIM (HLA Evolved Only)	4-9
4.2.7. Specifying the RPR FOM Version	4-10
4.2.8. Specifying the RPR FOM Revision	4-11
4.2.9. Ignoring Advisory Messages	4-11
4.2.10. Specifying a FOM Mapper	4-12
4.2.11. Configuring Time Management	4-14
4.2.12. Specifying the Logger Subscription List	4-15
4.2.13. Subscribing Passively to Objects and Interactions	4-18
4.2.14. Enabling Request Attribute Updates	4-18
4.2.15. Sending Only Updated Values	4-19
4.3. Configuring the Logger for DIS	4-20
4.3.1. Specifying a UDP Port	4-20

4.3.2. Setting the Exercise ID	4-21
4.3.3. Setting the Site ID	4-21
4.3.4. Setting the Application Number	4-21
4.3.5. Using Asynchronous I/O	4-22
4.3.6. Specifying a Destination Address for Outgoing and Incoming Data	4-22
4.3.7. Setting the Subnet Mask	4-23
4.3.8. Setting the Buffer Size for Sending and Receiving PDUs	4-23
4.3.9. Setting the Multicast Time to Live Value	4-24
4.3.10. Setting the Network Interface Address	4-24
4.3.11. Specifying the Heartbeat	4-25
4.3.12. Specifying the Timeout	4-26
4.3.13. Specifying the DIS Version	4-26
4.3.14. Subscribing to Multicast Groups	4-27

Chapter 5. Configuring the Logger

5.1. Configuring the Logger	5-2
5.2. Configuring Indexing	5-3
5.2.1. Configuring Indexing Using the Command Interfaces	5-4
5.3. Configuring Faster Playback	5-5
5.3.1. The SetEntityTracking Command	5-6
5.3.2. The SetFastJumpsWithoutDiff Command	5-6
5.3.3. The SetPlaybackReadTimeLimit Command	5-7
5.3.4. Configuring HLA Heartbeating	5-8
5.3.5. Configuring HLA Timeouts	5-9
5.4. Setting the Dead Reckoning Algorithm on Pause	5-9
5.5. Configuring Network Read Time	5-10
5.6. Using Data Distribution Management	5-11
5.6.1. Specifying Subscription Regions	5-12
5.6.2. Using Conveyed Regions	5-14
5.6.3. Using Simulation Data	5-15
5.6.4. Configuring DDM	5-15
5.6.5. Configuring DDM in <i>lgrConfig.xml</i>	5-17
5.7. Setting the Notification Level	5-18
5.7.1. Specifying the Log File	5-18
5.8. Editing <i>lgrConfig.xml</i>	5-19
5.8.1. Adding a Command to <i>lgrConfig.xml</i>	5-22
5.9. Summary of Command-line Options for the Logger	5-24

Chapter 6. Starting the Logger

6.1. Starting the Logger	6-2
6.1.1. Starting the Logger on Linux	6-2
6.1.2. Starting the Logger in Windows	6-2
6.1.3. Running the Logger without a GUI	6-2
6.2. The Logger Window	6-3
6.2.1. The Timeline	6-4

6.2.2. Hiding Plug-in Views	6-5
6.2.3. Docking and Undocking GUI Plug-in Views and Toolbars ...	6-6
6.3. Connecting to the Simulation Network	6-7
6.3.1. Disconnecting from the Simulation Network	6-7
6.4. Opening Logger Files	6-7
6.4.1. Upgrading a Logger File's Format	6-8
6.4.2. Opening Incomplete Files	6-9
6.4.3. Closing a Logger File	6-9
6.5. Listening to the Network	6-9
6.6. Viewing and Editing Logger File Information	6-10
6.6.1. Displaying Information about a Logger File	6-10
6.6.2. Editing the Logger File Description	6-11

Chapter 7. Recording and Playing Logger Files

7.1. Recording Exercises	7-3
7.1.1. Recording Immediately (to a Temporary File)	7-3
7.1.2. Recording to a Specified File	7-3
7.1.3. Recording to Multiple Files	7-4
7.1.4. Pausing the Recording Process	7-6
7.1.5. Stopping the Recording Process	7-6
7.2. Saving Files	7-7
7.2.1. File Backups	7-7
7.2.2. Saving a File to a New Name	7-7
7.2.3. Saving a Portion of a File to a New File	7-7
7.3. Playing Logger Files	7-8
7.3.1. Playing a Logger File	7-8
7.3.2. Pausing Playback of a Logger File	7-8
7.3.3. Playing a Segment of a File	7-8
7.3.4. Looping Playback	7-10
7.3.5. Stopping Playback of a Logger File	7-10
7.3.6. Setting the Playback Speed	7-10
7.3.7. Using High Speed Playback	7-11
7.3.8. Playing a File While it is Being Recorded	7-12
7.4. Zooming the Timeline	7-13
7.4.1. Zooming the Timeline Using the Mouse	7-13
7.4.2. Setting the Zoom from the Set Zoom Dialog Box	7-14
7.4.3. Moving the Zoomed Selection	7-14
7.5. Moving the Timepoint to a Specific Time	7-15
7.5.1. Jumping to the Beginning or End of the Playback Range	7-15
7.5.2. Moving to a Specific Time in a File	7-15
7.5.3. Setting the Time Using the Timeline Cursor	7-15
7.6. Viewing Packet Histograms	7-16
7.6.1. Scaling the Packet Count View	7-17
7.7. Viewing Object Statistics	7-18

Chapter 8. Annotating Files

8.1. Annotating a Logger File	8-2
8.1.1. Adding an Annotation	8-2
8.1.2. Saving an Annotation File	8-4
8.1.3. Opening an Annotation File	8-5
8.1.4. Importing an Annotation File	8-5
8.1.5. Selecting an Annotation	8-5
8.1.6. Editing an Annotation	8-5
8.1.7. Deleting an Annotation	8-6
8.1.8. Moving the Timeline Cursor to the Start of an Annotation	8-6
8.1.9. Jumping to the Next or Previous Annotation	8-7
8.1.10. Zooming the Timeline to an Annotation	8-8
8.1.11. Bounding Playback to an Annotation	8-9

Chapter 9. Filtering Input and Output

9.1. Filtering Logger Input and Output	9-2
9.2. Filtering by Entity Type	9-2
9.2.1. Removing Entity Types from the Entity Types Lists	9-3
9.2.2. Configuring the List of Common Entity Types	9-3
9.3. Filtering DIS PDUs by PDU Kind	9-4
9.4. Including Custom PDUs in Filter Lists	9-5
9.5. Filtering DIS Messages by Entity ID	9-6
9.6. Filtering HLA Messages by Object Name	9-7
9.7. Filtering HLA Exercises by Class	9-9
9.8. Filtering VR-Forces Messages	9-10

Chapter 10. Analyzing HLA Data

10.1. The HLA Traffic Analyzer	10-2
10.2. Traffic Analyzer Data Views	10-2
10.2.1. Hiding the Attributes Pane	10-3
10.3. How the Traffic Analyzer Evaluates Data	10-3
10.3.1. Representing Complex Datatypes	10-4
10.3.2. Datatype Representations Included in the Traffic Analyzer	10-4
10.4. Viewing Object and Interaction Classes and Attributes	10-5
10.4.1. Viewing the Attributes of an Object Class	10-5
10.4.2. Viewing the Parameters of an Interaction Class	10-5
10.5. Viewing Object Instances	10-6
10.6. Viewing Interaction Instances	10-7
10.7. Configuring the HLA Traffic Analyzer	10-9
10.7.1. Changing the Data Delimiter	10-10
10.7.2. Mapping Datatypes	10-11
10.7.3. Configuring Attribute Display Options	10-11
10.7.4. Mapping Entity Type Enumerations to Strings	10-12
10.7.5. Removing Brackets from the Display of Array Types	10-12

Chapter 11. Editing Logger Files

11.1. Editing Files	11-2
11.1.1. Reverting to the Saved Version of a File	11-2
11.2. Clearing and Deleting Segments of a File	11-3
11.3. Merging Files	11-5
11.4. Concatenating Files	11-5
11.5. Trimming Files	11-6
11.6. Offsetting Files	11-6

Chapter 12. Exporting Logger Data

12.1. Exporting Data	12-2
12.2. Exporting Data to a Database	12-3
12.2.1. Configuring an ODBC Data Source	12-6
12.2.2. Exporting DIS Data to a Database	12-7
12.2.3. Exporting HLA Data to a Database	12-8
12.2.4. The Export Log	12-8
12.3. Configuring a Data Source for Microsoft Access	12-9
12.4. Installing MySQL and Configuring a Data Source	12-13
12.4.1. Installing MySQL and the MySQL ODBC Driver	12-13
12.4.2. Creating a MySQL Database	12-14
12.4.3. Configuring a Data Source for MySQL	12-14
12.5. Importing Logger Data into a Spreadsheet	12-17
12.5.1. Importing Data from MySQL to Microsoft Excel	12-17
12.5.2. Importing Data from Access to Excel	12-21

Chapter 13. Using Logger Commands

13.1. Using Logger Commands on the Command Line	13-2
13.2. Using Logger Commands in the Configuration File	13-2
13.3. Using Logger Commands in the Text Interface	13-3
13.3.1. Entering Commands	13-3
13.3.2. Displaying a List of Commands	13-4
13.4. Command Descriptions	13-4
13.4.1. Argument Types	13-5
13.4.2. System Domain Commands	13-6
13.4.3. Player Domain Commands	13-7
13.4.4. Recorder Domain Commands	13-9
13.4.5. Simulation Domain Commands	13-11
13.4.6. Editor Domain Commands	13-19
13.5. Command List	13-20

Chapter 14. Using the Logger Remote Control

14.1. The Logger Remote Control	14-2
14.2. Configuring Loggers to be Controlled Remotely	14-2
14.3. Starting the Logger Remote Control	14-3
14.4. Recording Exercises with the Logger Remote Control	14-3

14.5. Playing Exercises with the Logger Remote Control	14-5
14.6. Configuring the Logger Remote Control Connection	14-6
14.7. Specifying the Refresh Rate	14-6
14.8. Setting the Timebar Color	14-7
14.9. Exiting the Logger Remote Control and Connected Loggers	14-7
14.10. Logger Remote Control Command-line Options	14-8

Appendix A. Troubleshooting

A.1. Troubleshooting Logger Network Communications (DIS Only)	A-2
A.1.1. Using the Correct Broadcast Address and UDP Port	A-2

MÄK Products Glossary

Index

MÄK Data Logger Quick Reference Card



Preface

The MÄK Data Logger (Logger) records network activity of HLA or DIS exercises and plays it back for review. The Logger has a graphical user interface, a text interface, and an API.

This manual is for persons who will install the Logger, for persons who will record or play back simulations, and for persons who will develop plug-ins or applications using the Logger APIs. It assumes that you are familiar with your operating system and that you know how to work in your computer's windowing environment.

Please consult *MÄK Data Logger Release Notes* for your version of the Logger for release-specific information and updates that occurred after this manual went to press.

How the Manual is Organized

The manual is organized as follows:

Chapter 1, *Introduction*, describes the Logger's features.

Chapter 2, *Installing the Logger*, explains how to install the Logger and related software such as the FLEXlm license manager and an RTI.

Chapter 3, *Logger Concepts*, provides an overview of what the Logger does.

Chapter 4, *Configuring the Logger Connection*, explains how to configure the Logger for use with HLA or DIS.

Chapter 5, *Configuring the Logger*, describes how to configure the Logger for faster performance, how to configure HLA DDM, and how to edit the Logger configuration file.

Chapter 6, *Starting the Logger*, explains how to start the Logger, connect to an exercise, and open Logger files.

Chapter 7, *Recording and Playing Logger Files*, shows how to record and play back HLA/DIS network activity using the Logger's graphical user interface.

Chapter 8, *Annotating Files* explains how to create and edit annotations in Logger files.

Chapter 9, *Filtering Input and Output* explains how to use the Logger's filtering capabilities to limit the messages being sent or received.

Chapter 10, *Analyzing HLA Data* explains how to use the HLA Traffic Analyzer to understand the HLA data that the Logger is receiving or sending.

Chapter 11, *Editing Logger Files* explains how to edit Logger files, including cutting segments, inserting files, and offsetting packet timestamps.

Chapter 12, *Exporting Logger Data* explains how to export Logger files to other formats, such as text and to MySQL or Microsoft Access databases.

Chapter 13, *Using Logger Commands*, explains how to execute Logger commands from the command-line, from a console window, and from the configuration file.

Chapter 14, *Using the Logger Remote Control*, explains how to use the Remote Control Logger to control multiple Loggers from one location.

Appendix A, *Troubleshooting*, shows how to correct network problems that may prevent the Logger from communicating properly.

The *MÄK Products Glossary*, defines terms used in the manual.

The *MÄK Data Logger Quick Reference Card* lists command-line options, keyboard shortcuts, and the commands that you can use with the command-line interface.

Logger Documentation

The Logger documentation set is as follows:

- ♦ *MÄK Data Logger Users Guide* describes how to install and use the Logger. It is distributed as an online manual in Adobe Portable Document Format (PDF).
- ♦ API documentation and a class reference based on the header files is provided in HTML format.
- ♦ Release documentation consists of release notes provided as a PDF file.
- ♦ Online help.
- ♦ *MÄK Products Interoperability Guide* describes general issues of HLA and DIS interoperability and specific issues for running MÄK products together.

The PDF files are in the *./doc* directory.

MÄK Products

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

- ♦ **VR-Link® Network Toolkit.** VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications.

VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.

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

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

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

- The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- ♦ MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- ♦ VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- ♦ VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- ♦ VR-inTerra. VR-inTerra is a C++ API for adding terrain agility to applications. It can load, page, or stream terrain from a wide variety of formats or sources into a single, consistent run-time representation, consisting of a collision graph and vector network.
- ♦ DI-Guy™. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
 - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
 - DI-Guy Scenario™. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.
 - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
 - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.

- DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.

How to Contact Us

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

Telephone

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

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com
VR-Vantage support:	vrv-support@mak.com

Internet

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

Post

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

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

Document Conventions

This manual uses the following typographic conventions:

Monospaced	Indicates commands or values you enter.
Monospaced Bold	Indicates a key on the keyboard.
<i>Monospaced Italic</i>	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
{ }	Indicates required arguments.
[]	Indicates optional arguments.
	Separates options in a command where only one option may be chosen at a time.
()	In command syntax, indicates equivalent alternatives for a command-line option, for example, (-h --help).
/	Indicates a directory. Since MÄK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
<i>Italic</i>	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose File → Save .
	Click the icon to run a tutorial video in the default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the Logger home directory. For example, the directory *logger5.4/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

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

Third Party Licenses

MÄK software products may use code from third parties. This section contains the license documentation required by these third parties.

Boost License

VR-Link, and all MÄK software that uses VR-Link uses some code that is distributed under the Boost License. All header files that contain Boost code are properly attributed. The Boost web site is: www.boost.org.

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the “Software”) to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link and all MÄK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MÄK Products. MÄK has made no modifications to these libraries. For more information about these libraries please see the following web sites:

- ♦ The LGPL license is available at: <http://www.gnu.org/licenses/lgpl.html>.
- ♦ Information about IconV is at: <http://www.gnu.org/software/libiconv/>.
- ♦ Information about LibXML is at: <http://xmlsoft.org/>.



1. Introduction

This chapter describes the basic features of the Logger.

Logger Features.....	1-2
Logger Interfaces	1-3
Filtering	1-3
Annotations.....	1-3
Editing.....	1-4
Data Export.....	1-4
HLA Traffic Analyzer.....	1-4
The Remote Control Logger	1-4
Sample Logger Files.....	1-5
Sample Programs.....	1-6
Distributed Simulation Standards Supported.....	1-7
RTI Support.....	1-7
FOM Support	1-7

1.1. Logger Features

The Logger records network activity of HLA and DIS exercises¹ and plays it back for review.

DIS only

In record mode, the Logger monitors all DIS traffic on a specified UDP port and stores it in a file. In playback mode, the Logger transmits DIS data packets stored in a Logger file. When the playback scale is less than or greater than 1.0, the Logger scales the dynamic quantities stored in the PDUs, such as velocity, acceleration, and so on.

HLA only

The HLA version of the Logger performs recording and playback while acting as a separate federate. In record mode, by default, it subscribes to all objects, attributes, and interactions in the RPR FOM, and records updates, interactions, and object removals that it perceives through the RTI. It records values for all published attributes of an object by requesting attribute updates as it discovers each new object.

In playback mode, the HLA Logger acts as a federate that owns each of the objects for which updates appear in the file. The Logger sends interactions and updates only for classes that other federates are subscribed to. The Logger serves as a proxy simulator, answering requests for attribute updates and making objectRemove RTI calls when it encounters removals in the recorded file. In addition, just as in DIS, velocities, accelerations, and angular velocities are scaled in outgoing messages when the Logger's time scale is not equal to zero.

If possible, the Logger uses the same object names as those used in the original simulation. If this is not possible, for example, if another federate is using the name, the Logger lets the RTI assign a name for the object in question.

1. The HLA counterpart to a DIS exercise is a federation execution. For simplicity, we use the term exercise to represent both.

1.1.1. Logger Interfaces

The Logger has the following interfaces:

- ♦ A graphical user interface (GUI).
- ♦ A text-based interface.
- ♦ A remote control interface.
- ♦ APIs that let you control the Logger through plug-ins, external applications, or over a network.

These interfaces have nearly identical capabilities. (Some features are only available through the GUI; others through the Logger API.) Common features include:

- ♦ Start, stop, or pause when playing a file.
- ♦ Navigate to a specific time-point when playing a file.
- ♦ Control playback speed and direction.
- ♦ Load a specific file for playback.

You can use all the interfaces at the same time.

Localization

The Logger includes translation files that allow you to localize the Logger's GUI. Please see [“Localizing the Graphical User Interface,”](#) on page 2-9 for details.

1.1.2. Filtering

You can filter the messages that the Logger records and sends to reduce network traffic or focus on a particular set of objects or interactions. In HLA, you can specify which classes you want to subscribe to. For details, please see Chapter 9, [Filtering Input and Output](#).

1.1.3. Annotations

You can annotate Logger files. Point annotations identify a specific time in a file. Range annotations identify a segment of a file. You can easily zoom the display to a range annotation and can bind the playback segment to a range annotation. For details, please see Chapter 8, [Annotating Files](#).

1.1.4. Editing

The Logger lets you easily edit Logger files. It supports the following types of edits:

- ♦ Merge files.
- ♦ Concatenate files.
- ♦ Clear or delete a segment of a file.
- ♦ Trim dead space at the beginning of a file.
- ♦ Offset the timestamps of the packets in a file.

For details, please see Chapter 11, [Editing Logger Files](#).

1.1.5. Data Export

The Logger can export data to the following formats:

- ♦ Text.
- ♦ Logger file.
- ♦ MySQL database using ODBC driver.
- ♦ Microsoft Access database using ODBC driver.

For details, please see Chapter 12, [Exporting Logger Data](#).

Developers can add additional export formats using the Logger API.

1.1.6. HLA Traffic Analyzer

The HLA Traffic Analyzer provides details about the FOM you are using and displays the data for the current known object instances and recent interactions. By examining the classes, you can debug problems that you are experiencing in your federation, such as an entity's location not being within the area for the scenario or an interaction missing parameter values.

1.1.7. The Remote Control Logger

The Remote Control Logger lets you control one or more Loggers from one location. You can record data and play it back from the Remote Control Logger. The primary use for the Remote Control Logger is to provide one point of control when you are using multiple Loggers to record a large exercise.

The Remote Control Logger is protocol-independent. It can control DIS Loggers, HLA Loggers, or both.

1.1.8. Sample Logger Files

The Logger includes several sample files. The files names indicate which protocol they support, such as DIS, or HLAe (HLA Evolved):.

- ♦ *maklanddemo2013*. (DIS, HLA 1.3, 1516, HLAe) A complex scenario with land, air, surface, and sub-surface entities. This file uses the Makland database, which is distributed with VR-Forces and MÄK's 2D and 3D visualization products.
- ♦ *alaskaTour2014*. (DIS) The Alaska Tour scenario showcases a fixed wing aircraft flying through a mountain range on the coast of Alaska.
- ♦ *bombMission2013*. (DIS, HLA 1.3, HLAe)
- ♦ *hawaiiNaval2013*. (DIS) This scenario demonstrates naval warfare capabilities of VR-Forces in six separate vignettes:
 - Mine clearing.
 - Naval gunfire.
 - Sonar and sonobuoys.
 - Submarine maneuvers.
 - Ballistic missiles.
 - Ship-ship engagement.
- ♦ *hawaiiTour2013*. (DIS, HLA 1.3) The Hawaii Tour scenario showcases a helicopter flying around the island of Oahu
- ♦ *Raid2013*. (DIS, HLAe) This scenario simulates a warehouse raid in Honolulu, Hawaii. Soldiers approach the raid location by helicopter and boat. They disembark, and four two-man teams move through cover to the doors of the warehouse, while the rest of the soldiers stay back to cover the approach. After the soldiers disembark, the helicopters leave and fly a route nearby, waiting for the signal to come back. One version has view controls. Another xxxxx SensorFX.
- ♦ *ProceduralTerrainDemo*. (DIS) Flyover by an F-18 over the Himalayas as well as three Apache helicopters in separate terrains: one circumnavigating Mt Rainier and two flying over terrain populated with trees. This intended to show off the state of VT MÄK's procedural terrain. Terrain to load with: WorldProceduralTerrain.
- ♦ *vehicleLights2014*. (DIS) Demonstrates lighting effects and lights cast by a HMMWV on the Hawaii terrain.
- ♦ *VR-TheWorld_FirstExperienceDIS*. DIS and HLA 1.3 RPR FOM 1.0. This recording is for use with *VR-TheWorld First Experience Guide*.

i

Most of these files contain View Control messages, which cause MÄK's 3D visualization products to follow the action automatically, switching among many predetermined viewpoints. For information about manually controlling the eyepoint in this demo, please see the 3D product documentation.

1.1.9. Sample Programs

The Logger includes the following example programs:

- ♦ Simple examples of using the Logger API — `playFile`, `readFile`, `recordFile`, `writeFile`, and `exportFileToDatabase`. Compiled versions of these examples are provided in the `./bin` directory.
- ♦ Example plug-ins — `stateObserver` displays a graph of object statistics. `lgrDump` prints details of each packet to the console. `sampleFilter` shows how to filter out selected packets.
- ♦ An example Logger application - `loggerApp`. You can use this example as a starting point for changing and recompiling the Logger application.
- ♦ A remote control application — `lgrControl`. Compiled versions of the example are provided in the `./bin` directory.
- ♦ A DDM example — `sampleDdm` shows how to manipulate DDM regions using packet data.
- ♦ `lgrModSqlExporter` — An example that extends the database export to include the DIS Data PDU.
- ♦ `matrexTspiFomMapper` — A VR-Link FOM Mapper for the entity attributes of the MATREX FOM for Time and Space Position Information.

For more information about examples, please see the Examples page in the online API documentation.

1.2. Distributed Simulation Standards Supported

The Logger has built-in support for DIS and the HLA RPR FOM. Please see release documentation for a list of DIS versions, RPR FOM versions, and RTI versions supported by your release of the Logger.

1.2.1. RTI Support

The Logger supports the HLA RTI 1.3 specification, the IEEE 1516 specification (SISO Dynamic Link Compatibility HLA API <http://www.sisostds.org/index.cfm>), and HLA Evolved (IEEE 1516-2010). The SISO 1516 API is an alternative mapping of the IEEE 1516 specification to C++. It supports dynamic link compatibility between different RTI implementations.

1.2.2. FOM Support

The Logger provides native support for several versions of the RPR FOM. (Please see the *MÄK Data Logger Release Notes* for a list of supported versions.) The Logger comes with FED files for HLA 1.3, HLA 1516, and HLA Evolved. The default FED file for HLA 1.3 is *MAK-RPR2-1-1.fed*, which contains all of the RPR FOM 2.0 objects and interactions. For HLA 1516, the Logger uses *MAK-RPR2-1-1.xml*. By default, the Logger joins the federation execution called VR-Link, and uses this FED file, which is distributed with the Logger, and with other MÄK tools.

The HLA Evolved version of the Logger uses *RPR_FOM_v2.0_1516-2010.xml* with the desired extensions (*MAK-DIGuy-1_evolved.xml*, *MAK-LgrControl-1_evolved.xml*, *MAK-VRFExt-1_evolved.xml*).

Using FOMs other than the RPR FOM

For simple recording and playback, the Logger can use any FOM. If you want to use advanced Logger features with FOMs other than the RPR FOM, you can write a VR-Link FOM mapper to provide the information necessary to properly map between VR-Link's internal object model and your FOM. Please see *VR-Link Developer's Guide* for complete details about creating FOM mappers.



2. Installing the Logger

This chapter explains how to install the Logger and related software.

Installing the MÄK Data Logger	2-2
Installing the MÄK Data Logger on Windows	2-2
Installing the Logger on a Linux System	2-2
The Logger Directory Structure	2-3
Installing and Setting Up the MÄK License Manager	2-4
Specifying the License Server	2-5
Installing an RTI	2-7
Installing the MÄK RTI	2-8
Localizing the Graphical User Interface	2-9
Translating Other Interface Files	2-12
Applying the Language Files	2-13

2.1. Installing the MÄK Data Logger

A full Logger installation includes three components: the application, the License Manager software, and for HLA operation, an RTI (or Runtime Infrastructure.)

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

2.1.1. Installing the MÄK Data Logger on Windows

Note the following:

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

Windows versions of the Logger are provided as executable installer files on DVD, or as downloaded files. The installers are named to indicate the compiler used to build that version of the Logger and whether it is a 32 bit or 64 bit version.

- To install the Logger, run the installer. Follow the instructions in the installation wizard.

2.1.2. Installing the Logger on a Linux System

Linux versions of the Logger are provided as compressed tar files on DVD, or as downloaded files.

To install the Logger on Linux:

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

```
tar -vxzf application.tar.gz
```

where *application* is the product release identification.

2.2. The Logger Directory Structure

The Logger installation process creates a directory structure like the one shown in [Figure 2-1](#).

```

loggerx.x.x
├── bin64
├── data
├── doc
├── examples
├── include
├── lib64
├── logger_tapes
├── mkcmake
└── translations

```

Figure 2-1. Logger directory structure

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

Table 2-1: Logger directory structure

Directory	Contents
<i>bin64</i>	Executables and DLLs.
<i>data</i>	Subdirectories contain default configuration files and images used in the GUI.
<i>doc</i>	Documentation.
<i>examples</i>	Examples of the Logger API and the Logger File API.
<i>include</i> and <i>lib64</i>	Header files and libraries for the Logger API and Logger File API.
<i>logger_tapes</i>	Sample Logger files.
<i>mkcmake</i>	Build scripts.
<i>translations</i>	Translation files for localizing the Logger.

2.3. Installing and Setting Up the MÄK License Manager

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



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

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

- ♦ Windows:
ftp://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (for 32-bit systems)
ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- ♦ Linux:
ftp://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz (for 32-bit systems)
ftp://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

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

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

2.3.1. Specifying the License Server

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

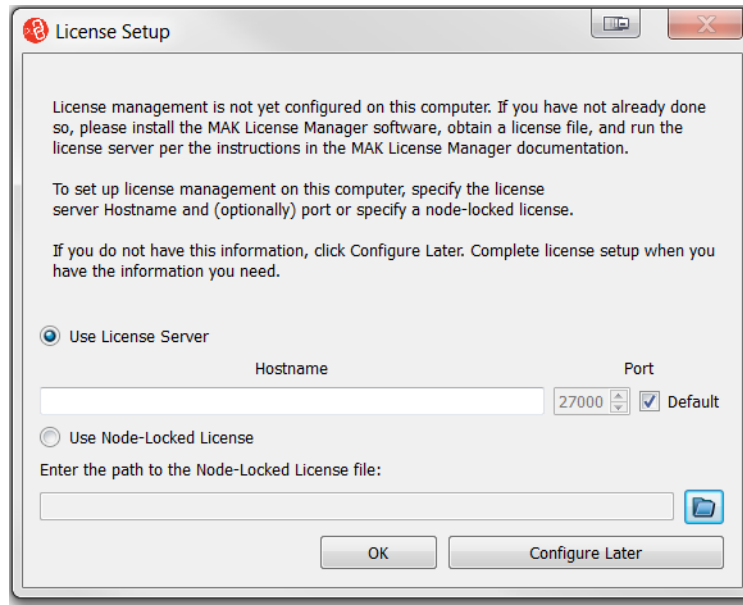


Figure 2-2. License Setup dialog box

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

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

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



- ♦ If you are running MAK products on the license server machine, it is also a client, so you must specify the license server on that machine too.
- ♦ If you change the license server, the saved configuration will no longer be valid and the License Setup dialog box will open the next time you start a MAK application.
- ♦ You can clear the saved license configuration by deleting the cache file.
On Windows 7, it is
`C:\Users\user_name\AppData\Roaming\MAK\licenses<n>.txt`. (The `AppData` directory is hidden by default.) On Linux, it is
`.mak/license<n>.txt`. (There may be more than one cache file, for example, `licenses1.txt` and `licenses2.txt`.)

The MAKLMGRD_LICENSE_FILE Environment Variable

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

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

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

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

1. Open the Control Panel.
2. On Windows 7, click System and Security.
3. Click System.
4. In the sidebar menu, click the Advanced System Settings. The System Properties dialog box opens.
5. Click Environment Variables. The Environment Variables dialog box opens.
6. Click New. The New System Variable dialog box opens.
7. In the Variable Name field, enter MAKLMGRD_LICENSE_FILE.
8. In the Variable Value field, enter `@server_name`, where `server_name` is the name of the license server.
9. Click OK to back out of each dialog box and set the variable.



Different versions of Windows have slightly different wording of the various Control Panel options.

Linux

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

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

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

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

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

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

2.4. Installing an RTI

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



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

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

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

2.4.1. Installing the MÄK RTI

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

Configuring Your System to Use the MÄK RTI

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

- ♦ Linux — LD_LIBRARY_PATH.
- ♦ Windows — PATH.

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

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

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

Running Applications with the MÄK RTI

To run a MÄK application with the MÄK RTI:

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

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

2.5. Localizing the Graphical User Interface

You can localize your MÄK product's graphical user interface so that it uses languages other than English. Your product includes a file called *.translations/product_untranslated.ts*, for example *logger_untranslated.ts* or *vrforces_untranslated.ts*. The file contains entries for all of the application-specific menu and dialog box text. The Qt Linguist tool, which is included with Logger, lets you map the English text provided by MÄK to your chosen language.

To edit a resource file:

1. Start the Qt Linguist utility. It is *.translations/linguist.exe*
2. In the Qt Linguist window, choose **File** → **Open**. Select the file *.translations/product_untranslated.ts*. The Settings for filename dialog box opens ([Figure 2-3](#)).

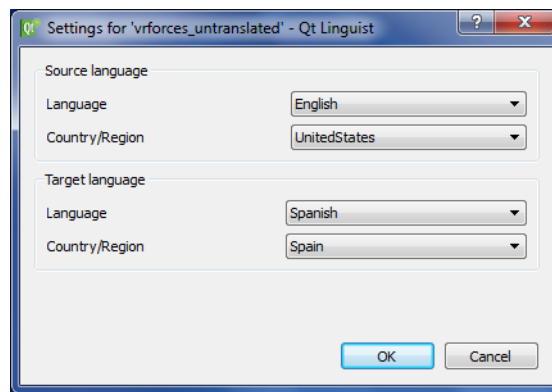


Figure 2-3. Linguist settings dialog box

3. Select values for the source language, target language, and target region or country.
4. Click OK. The translation file is opened and a list of classes is displayed on the left side of the window ([Figure 2-4](#)).

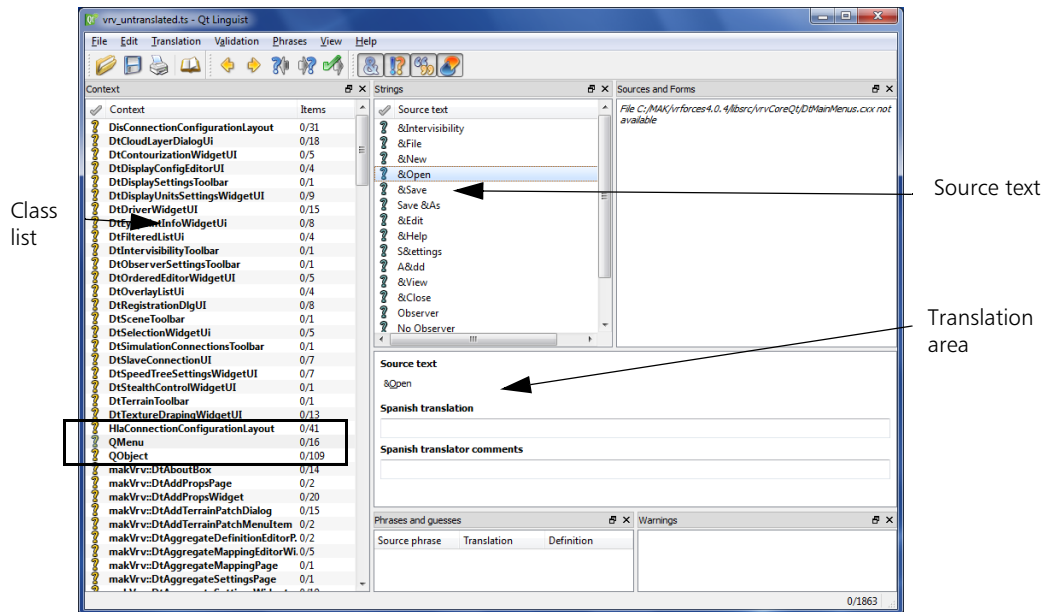


Figure 2-4. Linguist window

5. To translate text, you select each class in turn and translate the text associated with that class. You do not have to understand anything about what the classes mean, you just have to translate the text. In [Figure 2-4](#), *QMenu* is selected. The Strings window lists text that can be translated. Text that has not been translated has a question mark in the checkmark column.
6. In the Source Text list, select the word you want to translate. The text is displayed under Source Text in the translation area (the window below the source text list.) [Figure 2-5](#) shows the text *Open* selected.

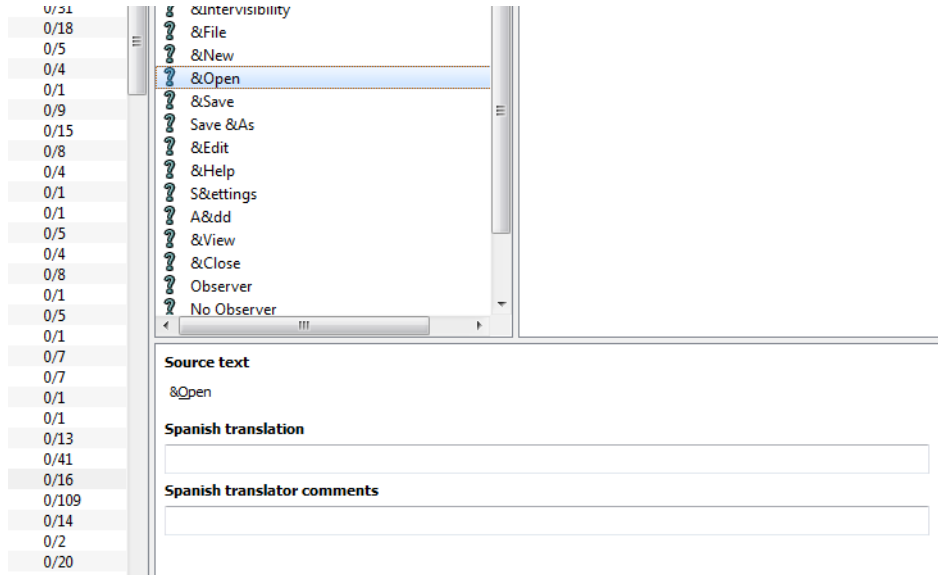


Figure 2-5. Text selected for translation

7. Place the cursor in the text box below *Language* Translation, where *Language* is the target language that you selected when you opened the translation file.
8. Type the translation text.
9. When you are done and you are certain of the translation, select the Done and Next button on the toolbar. A green check mark replaces the question mark (Figure 2-6). If you move to a different text string without clicking Done and Next, a yellow question mark is displayed for the string you translated. When you translate all elements for a class, a green check mark replaces the question mark next to the class name.

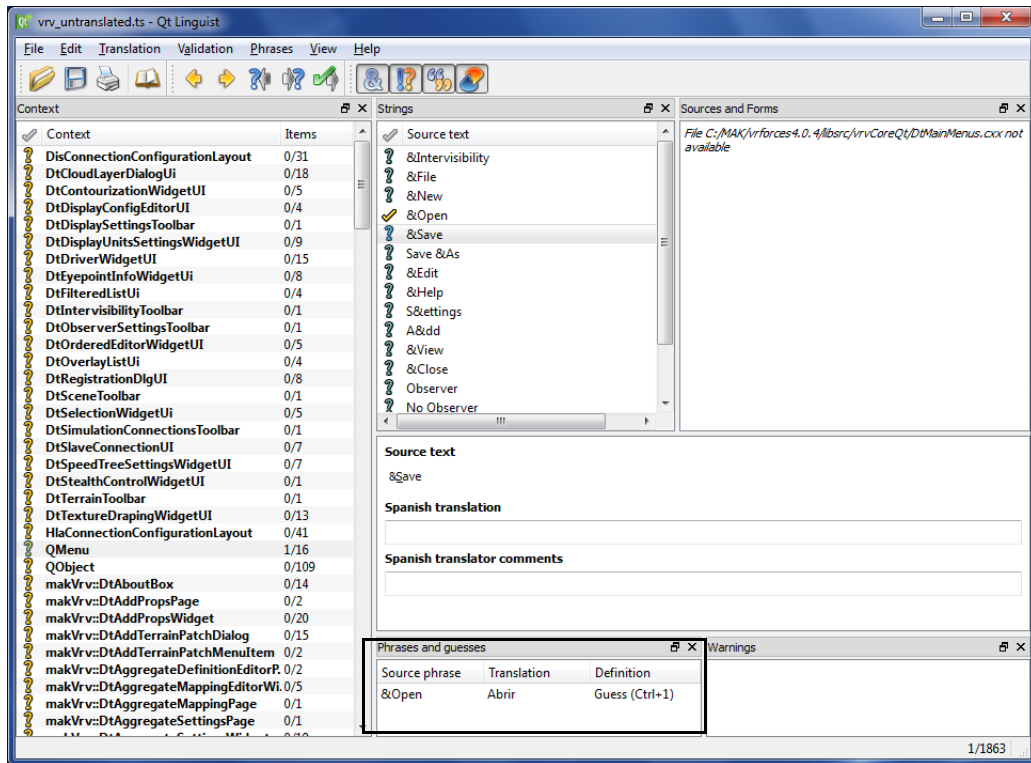


Figure 2-6. Completed translation for Open

10. Repeat this process for each element of source text for each class.
11. Choose **File** → **Release**. The file is saved in the format *filename.qm*.
12. Rename the file to a name that indicates the language to which you have translated the file, using the two-character abbreviation typically used with your language. For example, a Spanish translation would be renamed *product_es.qm*.
13. Save your translated *.ts* file.

2.5.1. Translating Other Interface Files

The *product_untranslated.ts* file contains all the text used in the Logger interface. However, there may be other text and messages that are not specific to the Logger that are generated by the software that implements the Logger GUI. To translate this text, you need to translate the file *qt_untranslated.ts*. Follow the same procedure as you did for *logger_untranslated.ts*.

2.5.2. Applying the Language Files

To put the language file into effect, use one of the following methods:

- ♦ Set your computer's environment to the proper language (LANG environment variable).
- ♦ Set the locale to the proper language.
- ♦ Start the Logger with the `-G locale` command line option, where *locale* is the two-letter country code for the local language.



3. Logger Concepts

This chapter describes the Logger's basic functioning.

What the Logger Does During Recording and Playback	3-2
The Exercise Connection.....	3-2
What The Logger Does During Recording.....	3-3
What The Logger Does During Playback.....	3-3
Recording and Playback Using HLA Time Management	3-4
Timescale During Playback.....	3-6
Using Time Scaling with Time Management	3-6
Compensating for Timescale Effects During Playback.....	3-6
HLA Support.....	3-8
FOM Flexibility	3-8
Backward Compatibility.....	3-8

3.1. What the Logger Does During Recording and Playback

This section describes, in general, what happens when you record and play back exercises. [Figure 3-1](#) illustrates the Logger window with a file loaded.

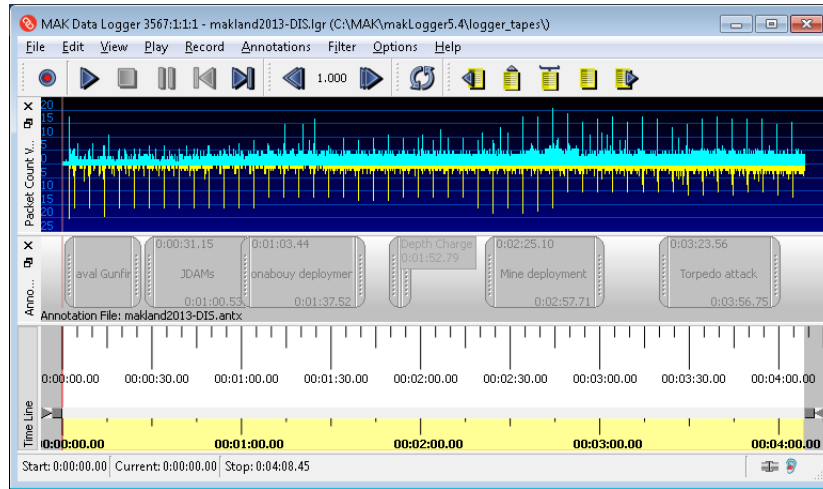


Figure 3-1. The Logger window

3.1.1. The Exercise Connection

Before you can record an exercise or play one back, you must connect to a DIS or HLA exercise. You can set connection parameters in *lgrConfig.xml*, from the command line, or through the GUI. For details, please see [“Configuring the Exercise Connection,”](#) on page 4-3.



Each version of the Logger has its own configuration file (*lgrConfigDIS.xml*, *lgrConfigHLA13.xml*, and so on). For simplicity this manual refers to it generically as *lgrConfig.xml* when the reference is not protocol-specific.

3.1.2. What The Logger Does During Recording

In record mode, the Logger is in either a recording state or paused state. When paused, the Logger does not record data and does not advance time.

DIS only

By default, the Logger connects to exercises using an exercise ID of 0, which means Any Exercise. When the exercise ID is 0, the Logger records all DIS packets it receives, without regard to exercise ID. With each PDU it records, the Logger stores a time-stamp indicating when the PDU occurred relative to the start of the file.

If you specify that the Logger connect with an exercise ID other than 0, it only records PDUs that have the same exercise ID.

HLA only

The Logger subscribes to objects, attributes, and interactions, and it records updates, interactions, and object removals that it perceives through the RTI. It requests updates as it discovers each new object, and records values for all published attributes of those objects.

If a recorded object has a name beginning with HLA_, the Logger adds LGR_ to the beginning of the name. This allows these objects to be properly published during playback. You can configure this behavior with the `SetDynamicNamePrefix` command.

3.1.3. What The Logger Does During Playback

When the Logger is in playback mode, it is in a playing state, a paused state, or a stopped state. When the Logger is playing, it reads messages from a file and transmits them in sequence over the network. When the Logger is paused, it halts at the current timepoint in the file and stops sending new data.

During playback, you can control the playback speed to produce real-time, fast motion, and slow-motion effects. For more information, please see [“Timescale During Playback,”](#) on page 3-6.

The Logger lets you navigate to any timepoint in a file. For procedures, please see [“Moving the Timepoint to a Specific Time,”](#) on page 7-15.

DIS only

When the Logger is paused, it repeatedly retransmits messages that represent the current state of each entity to maintain the DIS heartbeat. (This retransmission is not needed in HLA.)

In DIS, when you start the Logger and specify a file for playback, the Logger sends the recorded PDUs to the UDP port specified in *lgrConfigDIS.xml*, the command line, or in the Connection dialog box. If the Logger’s exercise ID is 0, it sends each PDU with the same exercise ID as the PDU had when it was recorded. If the Logger’s exercise ID is set to a number other than 0, all of the PDUs get sent with the Logger’s exercise ID.

HLA only

In HLA, the Logger acts as a federate that owns each of the objects represented in the file. The Logger sends interactions and updates only for classes that other federates are subscribed to. The Logger serves as a proxy simulator, answering requests for attribute updates and making objectRemove RTI calls when it encounters removals in the recorded file.



By default, HLA 1516 and HLA Evolved use name reservation. When name reservation is in effect, the RTI does not allow the Logger to create entities with names that have previously been used. This becomes an issue when you loop a Logger file. The RTI can rename objects the second time they are created, but this renaming does not get passed along to the attributes of objects such as aggregates or to interaction parameters. They will look for objects that have the original names and will not be able to find them.

If you are using the MÄK RTI, you can work around this problem by disabling the `RTI_enableNameReservation`, `RTI_strictNameReservation`, and `throwExceptionForDisabledService` parameters in *rid.mtl*.

3.1.4. Recording and Playback Using HLA Time Management

The Logger supports HLA Time Management in both record and playback mode. In record mode, the Logger participates as a time-constrained federate. In playback mode, it participates as a time-regulating federate. The contents of the files created in time-managed mode are identical to those created when not using Time Management, however the arrival time recorded for each packet is determined differently when using Time Management.



HLA Time Management is not supported in the text interface.

Playback Using Time Management

You can play any Logger file with or without Time Management enabled. If you play a file without using Time Management, packets are sent as RO updates. If you are using Time Management, they are sent as TSO updates.

To play files with Time Management enabled, you must use a FED file that is coded to indicate which FOM attributes and parameters are to be sent in TSO mode. The Logger comes with a properly written FED file named *VR-LinkTM.fed* for use with Time Management.

When the Logger joins a federation in time-managed mode, it asks to be a time-regulating federate. Its initial time request is the current value of the lower bound time stamp (LBTS). If there are no other time regulating federates (LBTS is infinity), the initial time request is zero.

The Logger starts playback at the point in the file where the arrival time equals or exceeds the current federation time grant plus the lookahead. Therefore, it is possible that the Logger could skip significant portions of the file, or even all of it. To ensure that you play the entire file, make sure that the Logger is the first time regulating federate to join the federation execution, or offset the times in the file relative to an appropriate starting time value.

Once it starts playback, the Logger advances at whatever rate is determined by the time scale setting. For more details, see [“Timescale During Playback,”](#) on page 3-6.

Some Logger Features are Restricted When You Use Time Management

When you play a file with Time Management enabled, the following restrictions apply:

- ♦ You cannot jump to different points in the file.
- ♦ You cannot pause playback.
- ♦ You can stop, but you cannot restart unless you restart the entire federation execution.

3.2. Timescale During Playback

The Logger lets you play a file at any speed within the capabilities of your platform.

You determine the speed by specifying a timescale value: the factor by which time is multiplied during playback. The default timescale setting of 1.0 causes a Logger file to play forward at normal speed. A timescale of 0.5 results in half-speed playback.

For procedures, please see [“Setting the Playback Speed,”](#) on page 7-10.

3.2.1. Using Time Scaling with Time Management

When you play a Logger file using Time Management, the Logger acts as a time regulating federate. If the Logger sends updates more slowly than other time regulating federates, it may slow down the federation execution. If it sends updates much more quickly than the other federates, it could overwhelm the RTI. Therefore, you should try to set the time scale to a value that is synchronized with the overall speed of federation execution.

3.2.2. Compensating for Timescale Effects During Playback

The Logger has corrective features that adjust the content of outgoing messages for different playback speeds. Under DIS, the Logger generates interim PDUs that prevent receiving applications from inappropriately timing out entities during slow-speed playback. Under both DIS and HLA, velocity and acceleration scaling can adjust message content to prevent anomalies in vehicle movement that would otherwise occur at any playback speed other than real-time.

Generating Interim PDUs During Slow-Speed DIS Playback (Heartbeating)

In a DIS exercise, each participating entity sends PDUs to other applications at a regular time interval, called the heartbeat. A common heartbeat rate is 5 seconds.

Most receiving applications expect to receive PDUs from each entity at the agreed-upon heartbeat rate. If a receiving application does not receive a PDU from an entity for an unusually long time, it may conclude that the entity no longer exists, and then stop simulating (or time out) the entity. A typical timeout threshold is 12 seconds, or 2.4 times the heartbeat rate.

During slow-speed playback in DIS, the long time lapse between an entity's recorded PDUs might cause receiving applications to time the entity out prematurely. To prevent this, the Logger generates interim PDUs as needed to maintain each entity's heartbeat.

The Logger calculates the contents of an interim PDU by applying a dead-reckoning algorithm to an entity's most recent PDU.

By default, the Logger ensures that at least one PDU is transmitted for each entity every five seconds. You can specify a different heartbeat rate. For procedures, please see [“Specifying the Heartbeat,”](#) on page 4-25.

Scaling the Timeout Period During Slow-Speed Playback

When an entity's PDUs stop occurring in the file, the Logger continues to generate interim PDUs for the entity for the specified timeout period.

During slow-speed playback, the timeout period is extended accordingly. For example, when playing a file at half-speed, using the default timeout setting of 12 seconds, the Logger generates interim PDUs for an entity for 12 recorded seconds, or 24 real-time seconds before timing it out.

For procedures, please see [“Specifying the Timeout,”](#) on page 4-26.

Updating Entity Information Quickly After a Time Jump

During playback, you can jump to an arbitrary timepoint. After the time jump, the Logger resumes sending update messages for all entities, starting at the new timepoint. Some time may pass before a fresh update message is sent for each entity, so until updates arrive, receiving applications will reflect entity states that existed before the time jump.

DIS only

In DIS, these applications may time entities out.

After a time jump, the Logger can generate up-to-date interim messages to fill the period before the transmission of each entity's next recorded update. To create these interim updates, the Logger applies a dead-reckoning algorithm to each entity's most recent recorded message prior to the new time point.

Because additional processing is required, you will experience a delay when jumping to a new timepoint. The length of the delay depends on the number of entities processed and the size of the time jump.

Velocity and Acceleration Scaling

When the Logger plays at speeds above or below normal, it can adjust the velocity and acceleration values in entity state messages. This scaling allows the receiving applications to perform accurate dead-reckoning.

For example, if the Logger is playing at double speed, it can double the values in a message's velocity and acceleration fields. This adjustment causes a vehicle to move at double its original speed when viewed using an application such as VR-Vantage Stealth. Without this scaling, a moving vehicle would appear to be moving at its original slower speed, and would have to lurch forward with the arrival of each new state update message in order to catch up with the new location data contained in the update.

3.3. HLA Support

This section describes compatibility and support issues for running the Logger in HLA. Please refer to *MÄK Data Logger Release Notes* and any supplementary documentation for your version of the Logger for current information about FOM support.

3.3.1. FOM Flexibility

You can change the FED file between recording and playback. As long as the FED file you are using on playback contains object and interaction classes, and attributes and parameters with the same names and format as those that were recorded, you can play back your file.

Classes, attributes, and parameters can be rearranged in the FED file, and new ones can be added. If any class names have been deleted from the FED file between recording and playback, recorded interactions and updates for those classes will not be included in the Logger's outgoing data stream. If any attributes or parameter names have been deleted from the FED file, those attributes and parameters do not appear in outgoing updates and interactions.

3.3.2. Backward Compatibility

Different versions of the RPR FOM represent many attributes and parameters differently. They are essentially different protocols. Therefore, you may not be able to play back a Logger file recorded with previous versions of the Logger in the same federation execution as applications based on the current RPR FOM.

Please refer to the *MÄK Data Logger Release Notes* and any supplementary documentation for the latest release of the Logger for current information about backward compatibility.



4. Configuring the Logger Connection

This chapter explains how to configure connections for DIS and HLA. For general information about configuring the Logger, please see Chapter 5, *Configuring the Logger*.

Configuring the Exercise Connection	4-3
Viewing Connection Properties.....	4-3
Editing a Connection	4-4
Configuring the Logger for HLA	4-6
Specifying the Name of the Logger Federate (HLA Evolved Only).....	4-7
Specifying the Type of the Logger Federate	4-7
Specifying the Federation Execution Name	4-8
Specifying a FED File.....	4-8
Specifying FOM Modules (HLA Evolved Only)	4-9
Specifying the MIM (HLA Evolved Only)	4-9
Specifying the RPR FOM Version	4-10
Specifying the RPR FOM Revision	4-11
Ignoring Advisory Messages.....	4-11
Specifying a FOM Mapper.....	4-12
Configuring Time Management.....	4-14
Specifying the Logger Subscription List.....	4-15
Subscribing Passively to Objects and Interactions.....	4-18
Enabling Request Attribute Updates	4-18
Sending Only Updated Values	4-19
Configuring the Logger for DIS.....	4-20
Specifying a UDP Port	4-20
Setting the Exercise ID	4-21
Setting the Site ID.....	4-21
Setting the Application Number.....	4-21

Configuring the Logger Connection

Using Asynchronous I/O.....	4-22
Specifying a Destination Address for Outgoing and Incoming Data.....	4-22
Setting the Subnet Mask	4-23
Setting the Buffer Size for Sending and Receiving PDUs.....	4-23
Setting the Multicast Time to Live Value	4-24
Setting the Network Interface Address	4-24
Specifying the Heartbeat	4-25
Specifying the Timeout	4-26
Specifying the DIS Version	4-26
Subscribing to Multicast Groups.....	4-27

4.1. Configuring the Exercise Connection

The Logger connects to exercises through an exercise connection. You can connect to and disconnect from an exercise. When you connect to an exercise you must specify the parameters for the protocol you are using (DIS or HLA). The connection status is displayed in the Logger status bar (Figure 4-1).

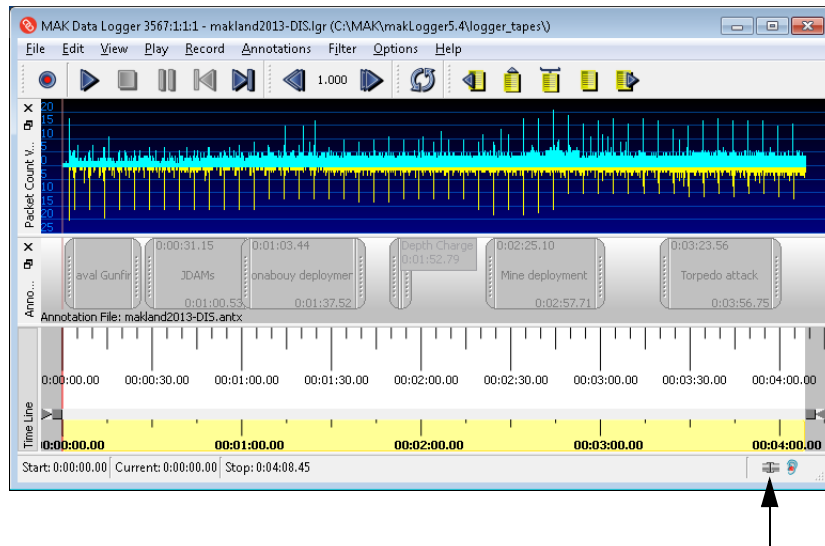


Figure 4-1. Connection status

4.1.1. Viewing Connection Properties

In the GUI, you can view the configuration of the current exercise connection for the Logger. When the Logger is connected to an exercise, the connection settings are read-only.

- To view connection properties, choose **File** → **Connection**, or right-click the connection status icon. The Connection dialog box opens (Figure 4-2 and Figure 4-3).

4.1.2. Editing a Connection

If you are not connected to an exercise, you can set connection properties and connect to the network. For descriptions of the various properties, please see “Configuring the Logger for HLA,” on page 4-6 and “Configuring the Logger for DIS,” on page 4-20.

By default, when you change connection properties, they are saved to the *lgrConfig.xml* configuration file. You can enable and disable this setting on the Preferences dialog box, Network Settings page.



You cannot edit a connection while you are connected. To make a connection editable, disconnect from it.

To edit a connection:

1. If the Logger is connected to an exercise, choose **File** → **Disconnect**.
2. Choose **File** → **Connection**. The Connection dialog box opens. It displays the current settings for the protocol you are using (HLA (Figure 4-3) or DIS (Figure 4-3)).

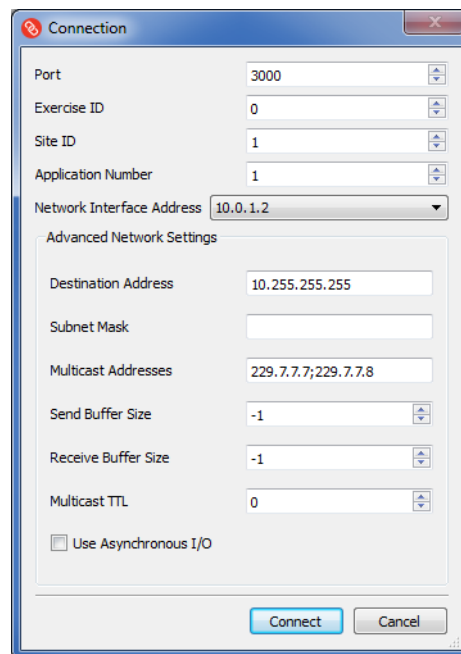


Figure 4-2. Connection dialog box for DIS

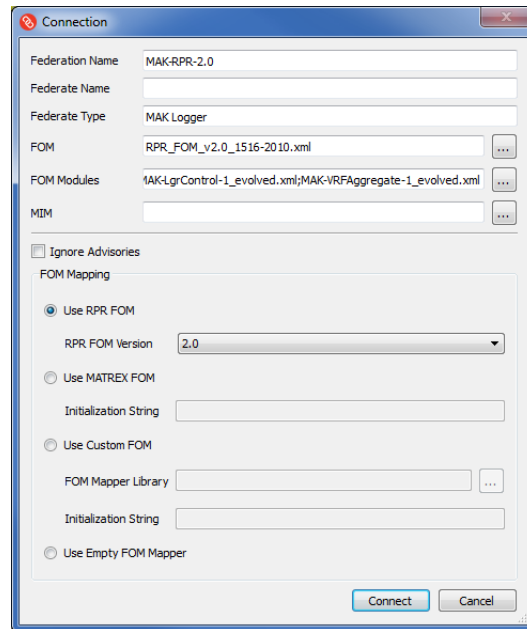


Figure 4-3. Connection dialog box for HLA Evolved

3. Change attribute values as desired.
4. Click Connect.

4.2. Configuring the Logger for HLA

This section describes how to configure the Logger for HLA exercises. For details about configuring connection settings in the GUI, please see [“Editing a Connection,”](#) on page 4-4.

Most of the HLA connection settings can be set using individual command-line options or by using the `SetConnectionParameters` command in the text interface.

The syntax for `SetConnectionParameters` for HLA 1.3 and HLA 1516 is:

```
SetConnectionParameters federateName federationName
    fedFile ignoreAdvisories fomVersion fomRevision
    fomMapper fomMapperData useEmptyFomMapper
```

For example:

```
SetConnectionParameters "MAK Logger" MAK-RPR2.0
    MAK-RPR2-1-1.fed False 2.0 1 myFomMapper
    "myInitParameter1 myInitParameter2 ..." False
```

The syntax for `SetConnectionParameters` for HLA Evolved is:

```
SetConnectionParameters federateName federateType
    federationName fedFile "fomModules" MIM ignoreAdvisories
    fomVersion fomRevision fomMapper fomMapperData
    useEmptyFomMapper
```

For example:

```
SetConnectionParameters Logger "MAK Logger" MAK-RPR2.0
    MAK-RPR2-1-1.xml "VR-Link_evolved_module.xml extendF18-
    Module.xml" HLAstandardMIM.xml False 1.0 1 myFomMapper
    "myInitParameter1 myInitParameter2 ..." False
```

The `fomMapperData` argument can be a list. If so, you must wrap it in quotes.

When you use the `SetConnectionParameters` command, you must specify all of the arguments, whether you want to change them from the default values or not. For example, if you just want to change the name of the Logger federate, even though it is the first argument in the list, you still must enter a complete command as follows:

```
SetConnectionParameters myLogger MAK-RPR2.0
    MAK-RPR2-1-1.fed "" 2.0 1 "" "" ""
```

For details about commands and command syntax, please see Chapter 13, [Using Logger Commands](#).

4.2.1. Specifying the Name of the Logger Federate (HLA Evolved Only)

If you are using HLA Evolved, you must specify the federate name. By default, the Logger federate is named MAK Logger.

- To specify the federate name, use the (`-N` | `--federateName`) command-line option or the `SetConnectionParameters` command.

Table 4-1: Specifying the federate name

Interface	Command
GUI	Connection dialog box: Federate Name attribute.
Command line	<code>loggerversion -N myLogger</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>IgrConfig.xml</i>	The <code>federateName</code> variable is in the default configuration file.

4.2.2. Specifying the Type of the Logger Federate

By default, the Logger federate type is “MAK Logger”.

- To specify the federate type, use the `--federateType` command-line option or the `SetConnectionParameters` command.

Table 4-2: Specifying the federate type

Interface	Command
GUI	Connection dialog box: Federate Type attribute.
Command line	<code>loggerversion --federateType "my custom Logger"</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>IgrConfigHLA1516e.xml</i>	The <code>federateType</code> variable is in the default configuration file.

4.2.3. Specifying the Federation Execution Name

By default, the Logger uses the federation execution name MAK-RPR-2.0.



By default, the Logger assumes that the federation configuration file (FED, FDD) file has the same name as the federation execution, plus the appropriate extension. For details, please see [“Specifying a FED File,”](#) on page 4-8.

- To specify a federation execution name, use the `(-x | --execName)` command-line option or the `SetConnectionParameters` command.

Table 4-3: Specifying the federation execution name

Interface	Command
GUI	Connection dialog box: Federation Name attribute.
Command line	<code>loggerversion -x MAK-RPR-2.0</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>execName</code> variable is in the default configuration file.

4.2.4. Specifying a FED File

By default the Logger uses a FED file called *MAK-RPR2-1-1.fed*. The Logger also accepts federation description files with the *.fdd* and *.xml* extensions. MAK products come with a FED (and XML) file for each version of the RPR FOM that they support.



If you are using HLA Evolved, the FED (FDD) file defines the primary FOM, to which FOM modules may be added.

- To specify a FED file, use the `(-F | --fedFileName)` command-line option or the `SetConnectionParameters` command.

Table 4-4: Specifying the FED file

Interface	Command
GUI	Connection dialog box: FED File Name attribute.
Command line	<code>loggerversion -F MAK-RPR-2.0.fed</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>fedFileName</code> variable is in the default configuration file.

4.2.5. Specifying FOM Modules (HLA Evolved Only)

If you are using HLA Evolved, you can specify FOM modules. FOM modules get added to the primary FOM, which is specified by the FED file. By default, the Logger configures FOM Modules in *lgrConfigHLA1516e.xml*. The Logger also accepts FOM Module files with the *.xml* extension. MÄK products come with a few FOM modules that define the VR-Link specific classes that are used by our products.

- To specify FOM modules, use the `--fomModules` command-line option or the `SetConnectionParameters` command.

Table 4-5: Specifying FOM modules

Interface	Command
GUI	Connection dialog box: FOM Modules attribute.
Command line	<code>loggerversion --fomModules "module1.xml module2.xml ... modulen.xml"</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfigHLA1516e.xml</i>	The <code>fomModules</code> variable is in the default configuration file.

4.2.6. Specifying the MIM (HLA Evolved Only)

If you are using HLA Evolved, you can specify a MOM and Initialization Module (MIM). By default the Logger does not use a MIM. The Logger accepts MIM files with the *.xml* extension. MÄK products come with the standard MIM.

- To specify the MIM, use the `--MIM` command-line option or the `SetConnectionParameters` command.

Table 4-6: Specifying the MIM

Interface	Command
GUI	Connection dialog box: MIM attribute.
Command line	<code>loggerversion --MIM HLAstandardMIM.xml</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfigHLA1516e.xml</i>	The <code>MIM</code> variable is in the default configuration file.

4.2.7. Specifying the RPR FOM Version

The Logger has built-in FOM Mapping support for the RPR FOM 1.0 and RPR FOM 2.0 standards as well as many of the draft RPR FOM versions (0.5, 0.7, 0.8, 2.006, 2.0014, and 2.0017). For each RPR FOM version, there is a separate FOM Mapper. All the FOM Mappers for the RPR FOM are contained in a VR-Link library used by the Logger. When you select the FOM Mapping to be RPR FOM, you must also choose which of the RPR FOM Mappers to use. The default version is 2.0. If you specify a RPR FOM version other than the default, you must also specify the correct FED file.



Please see *MÄK Data Logger Release Notes* for a list of the RPR FOM versions supported by your version of the Logger.

- To specify the RPR FOM version, use the `--rprFomVersion` command-line option or the `SetConnectionParameters` command.

Table 4-7: Specifying the RPR FOM version

Interface	Command
GUI	Connection dialog box: Use RPR FOM option and RPR FOM Version attribute.
Command line	<code>loggerversion --rprFomVersion 2.0017 -F VR-Link20017-1.fed</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>rprFomVersion</code> variable is in the default configuration file.

4.2.8. Specifying the RPR FOM Revision

Each RPR FOM version has a separate RPR FOM Mapper. However, in some cases the implementation of a particular RPR FOM Mapper has required modification to correct misinterpretations of the standard. To support backwards compatibility with VR-Link versions that contain the misinterpretation, these modifications are associated with a revision number. The revision number can then be used by a newer RPR FOM Mapper to modify its encodings and decodings to revert back to an older VR-Link version that does not support the newer corrected interpretations. The default revision is 1. If you specify a RPR FOM revision other than the default, you must also specify the correct FED file, which may or may not be the same as the current selection. For example, the FOM Mapper for RPR FOM 2 draft 17 revisions 1 through 5 use a FED or XML file named *MAK-RPR20017-1-1*; revision 6 uses *MAK-RPR20017-6-1*. There are versions for HLA Evolved with a similar naming convention.

- To specify the RPR FOM revision, use the `--rprFomRevision` command-line option or the `SetConnectionParameters` command.

Table 4-8: Specifying the RPR FOM revision

Interface	Command
GUI	Connection dialog box: Use RPR FOM option and RPR FOM Revision attribute.
Command line	<code>loggerversion --rprFomVersion 2.0017 --rprFomRevision 1 -F VR-Link20017-1.fed</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>rprFomRevision</code> variable is in the default configuration file.

4.2.9. Ignoring Advisory Messages

- To have the Logger ignore advisory messages, use `--ignoreAdvisories` on the command line or use the `SetIgnoreAdvisories` command in the text interface or configuration file.

Table 4-9: Specifying the response to advisory messages

Interface	Command
GUI	Connection dialog box: Select Ignore Advisories.
Command line	<code>loggerversion --ignoreAdvisories 1</code>
Text interface	Use the <code>SetConnectionParameters</code> command, or use <code>SetIgnoreAdvisories true</code>
<i>lgrConfig.xml</i>	The <code>ignoreAdvisories</code> variable is in the default configuration file. You can also add <code>SetIgnoreAdvisories</code> to the configuration file.

4.2.10. Specifying a FOM Mapper

The Logger supports different FOMs using VR-Link FOM Mappers. The Logger chooses the FOM Mapper based on the following criteria:

- If a FOM mapper library is specified, it is used.
- If Use Empty FOM Mapper is specified, the empty FOM Mapper is used.
- If neither of the first two options is specified, a default RPR FOM FOM Mapper is used.



The different versions of the RPR FOM that you can specify with the `rprFomVersion` parameter are actually designating one of several default FOM Mappers supplied with the Logger.

- To specify a FOM Mapper, use the `(-f | --fomMapperLib)` or `--useEmptyFomMapper` command-line option or the `SetConnectionParameters` command.

Table 4-10: Specifying a FOM Mapper

Interface	Command
GUI	Connection dialog box: Select Use Custom FOM and specify FOM Mapper Library and, optionally, Initialization String. Select Use Empty FOM Mapper if the FOM you are using is not a derivative of the RPR FOM and no appropriate FOM Mapper library is available.
Command line	<code>loggerversion -f myFomMapper.dll</code> or <code>loggerversion --useEmptyFomMapper 1</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>fomMapperLib</code> and <code>useEmptyFomMapper</code> variables are in the default configuration file.

Specifying a FED File that is Appropriate for the FOM Mapper



The `-f` option only tells the application which FOM Mapper to use. It does not ensure that you are using the correct FED file for the FOM Mapper. Make sure that you are using a FED file that is consistent with the FOM Mapper you have chosen.

By default the Logger uses a FED file called *MAK-RPR2-1-1.fed*, but you can specify an alternate FED file using the `-F` option.

The following example instructs the application to use an alternative FOM Mapper and FED file:

```
loggerversion -f myFomMapper -F myFedFile.fed
```

Specifying FOM Mapper Initialization Data

Some FOM Mappers require initialization data.

- To specify FOM Mapper initialization data, use the `--fomMapperData` command-line option or the `SetConnectionParameters` command.

Table 4-11: Specifying FOM Mapper initialization data

Interface	Command
GUI	Connection dialog box: Set Use FOM Mapper to True, specify a FOM Mapper File, and set the FOM Mapper Init Data attribute.
Command line	<code>loggerversion -f myFomMapper.dll --fomMapperData ABC</code> where ABC is passed to the FOM Mapper creator function in the shared library as an initialization parameter.
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>fomMapperData</code> variable is in the default configuration file.

4.2.11. Configuring Time Management

The Logger supports time-constrained recording and time-regulating playback. To configure time management in the Logger, it must be enabled in your RTI. If time management is not enabled in the RTI, the configuration options are not available in the Logger Preferences dialog box.

To configure time management in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Configuration page (Figure 4-4).

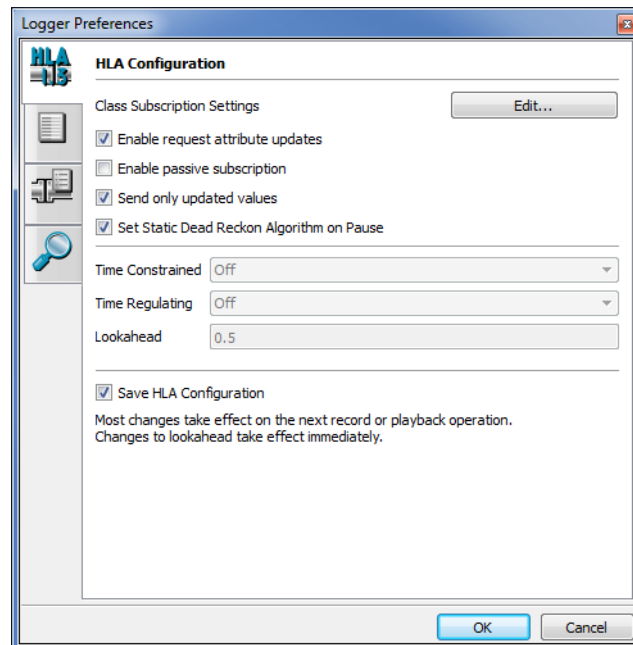


Figure 4-4. Logger Preferences dialog box, HLA Configuration page

3. To configure the Logger as a time-constrained federate, set the Time Constrained value to Recording.
 4. To configure the Logger as a time-regulating federate, set the Time Regulating value to Playback.
 5. Optionally, set the Lookahead value.
 6. Click OK.
- To specify time management from a command interface, use the `SetTimeManagement` command.

Table 4-12: Specifying time management

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "SetTimeManagement true false false true 0.5"</code>
Text interface	<code>SetTimeManagement true false false true 0.5</code>
<i>lgrConfig.xml</i>	Add <code>SetTimeManagement</code> to the configuration file.

4.2.12. Specifying the Logger Subscription List

By default the Logger subscribes to all classes in your FED file except *RTIPrivate* and any MOM object or interaction. However, you can specify the set of classes and interactions that the Logger subscribes to by listing them in a command or creating a subscription control list.

Creating a Subscription Control List

In the Logger GUI, you can specify a list of classes to subscribe to and you can save that list to a file.

To create a subscription control list in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Configuration page (Figure 4-4).
3. On the Class Subscription Settings line, click Edit. The Change HLA Class Subscription dialog box opens (Figure 4-5). The dialog box has a tab that lists object classes and a tab that lists interaction classes. Classes that have a check mark next to their name are filtered out.

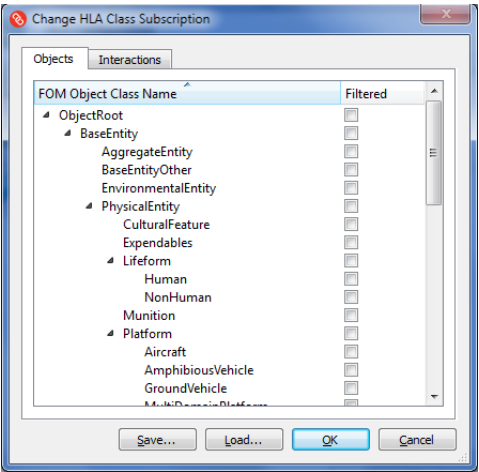


Figure 4-5. Change HLA Class Subscription dialog box

- 4. Edit the subscription list by checking the box in the Filtered column for classes that you want to filter out.
- 5. Click OK. The subscription list takes effect the next time you enter record mode.
- To specify a subscription list in a command interface, use the `SetSubscriptionLists` command.

Table 4-13: Specifying subscription lists

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "SetSubscriptionLists \"Object-Root.BaseEntity.PhysicalEntity.Platform.Aircraft Object-Root.BaseEntity.PhysicalEntity.Radio InteractionRoot.WeaponFire\""</code>
Text interface	<code>SetSubscriptionLists "ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft ObjectRoot.BaseEntity.PhysicalEntity.Radio InteractionRoot.WeaponFire"</code>
<i>lgrConfig.xml</i>	Add <code>SetSubscriptionLists</code> to the configuration file. For details about adding a string list to the configuration file, please see "Adding a String List to lgrConfig.xml," on page 5-23.
	! The <code>SetSubscriptionLists</code> command must come after the <code>Connect</code> command.

Saving A Subscription List

To save a subscription list in the GUI:

1. Edit a subscription list as described in [“Specifying the Logger Subscription List,”](#) on page 4-15.
 2. In the Change HLA Class Subscription dialog box, click Save. The Save As dialog box opens.
 3. Enter a name for the subscription list and click Save. By default, subscription lists are saved in the *bin* directory with the extension *.subx*.
- To save a subscription list in a command interface, use the `SaveSubscriptionListFile` command.

Table 4-14: Saving a subscription list

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "SaveSubscriptionListFile mySubList.subx"</code>
Text interface	<code>SaveSubscriptionListFile mySubList.subx</code>

Loading A Subscription List

To load a subscription list in the GUI:

1. In the Change HLA Class Subscription dialog box, click Load. An Open dialog box opens.
 2. Select the subscription list file you want to load.
 3. Click Open.
 4. Click OK to apply the subscription list.
- To load a subscription list in a command interface, use the `LoadSubscriptionListFile` command.

Table 4-15: Loading a subscription list

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "LoadSubscriptionListFile \"../config/mySubList.subx\""</code>
Text interface	<code>LoadSubscriptionListFile ../config/mySubList.subx</code>
<i>lgrConfig.xml</i>	Add <code>LoadSubscriptionListFile</code> to the configuration file.

4.2.13. Subscribing Passively to Objects and Interactions

The Logger can subscribe passively to objects and interactions. Passive subscriptions instruct the RTI to reflect any matching updates that are sent, but it will not trigger RTI advisories such as `startRegistration` or `turnUpdatesOn`. In this mode, the Logger will not cause any updates to be sent based on its joining the federation. As a result, the Logger should join the federation before any critical updates or interactions that need to be recorded are sent. If you want to use passive subscription, you must specify it before you connect to an exercise.

To configure passive subscription:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
 2. Select the HLA Configuration page ([Figure 4-4](#)).
 3. Select or clear the Enable Passive Subscription check box.
 4. Click OK.
- To have the Logger subscribe passively, use the `SetPassiveSubscription` command.

Table 4-16: Specifying passive subscription

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "SetPassiveSubscription true"</code>
Text interface	<code>SetPassiveSubscription true</code>
<i>lgrConfig.xml</i>	Add <code>SetPassiveSubscription</code> to the configuration file.

4.2.14. Enabling Request Attribute Updates

By default, the Logger makes `requestAttributeValue` update calls when it discovers objects. You can enable or disable this behavior.

To configure whether the Logger requests attribute value updates in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Configuration page ([Figure 4-4](#)).
3. Select or clear the Enable Request Attribute Updates check box.
4. Click OK.

- To configure whether the Logger requests attribute value updates when it discovers an object, use the `EnableRequestUpdates` command.

Table 4-17: Specifying attribute value update requests

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "EnableRequestUpdates true"</code>
Text interface	<code>EnableRequestUpdates true</code>
<i>lgrConfig.xml</i>	Add <code>EnableRequestUpdates</code> to the configuration file.

4.2.15. Sending Only Updated Values

During playback, the Logger may receive attribute value update requests for attribute instances that either are not in the file or not played back yet. You can specify that the Logger only send updates for values that it has retrieved from the file. If you disable this option, the Logger sends default values.



When changing scale or for heartbeating, the Entity and Aggregate publishers send all data. This data may include attributes that have not been recorded in the file.

To configure how the Logger responds to attribute value update requests in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
 2. Select the HLA Configuration page (Figure 4-4).
 3. Select or clear the Send Only Updated Values check box.
 4. Click OK.
- To configure how the Logger responds to attribute value update requests with the command interface, use the `SetSendOnlyUpdatedValues` command.

Table 4-18: Specifying attribute value update requests

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerversion -r "SetSendOnlyUpdatedValues true"</code>
Text interface	<code>SetSendOnlyUpdatedValues true</code>
<i>lgrConfig.xml</i>	Add <code>SetSendOnlyUpdatedValues</code> to the configuration file.

4.3. Configuring the Logger for DIS

This section describes how to configure the Logger for DIS exercises.

Most of the DIS connection settings can be set using individual command-line options or by using the `SetConnectionParameters` command in the text interface. The syntax for this command is:

```
SetConnectionParameters port exerciseId siteId appNumber
                        useAsynchIO sendBufferSize recvBufferSize multicastTtl
                        destAddr subnetMask deviceAddr mcastAddresses
```

For example:

```
SetConnectionParameters 3000 1 1 2 True -1 -1 -1
10.255.255.255 "" 10.0.1.57 "225.0.0.118 225.0.0.202"
```

When you use the `SetConnectionParameters` command, you must specify all of the arguments, whether you want to change them from the default values or not. For example, if you just want to change the port, even though it is the first argument in the list, you still must enter a complete command as follows:

```
SetConnectionParameters 3000 1 1 2 True -1 -1 -1
10.255.255.255 "" 10.0.1.57 "225.0.0.118 225.0.0.202"
```



The Logger supports IPv4 and IPv6. For any particular connection, all addresses must be of the same type.

For details about commands and command syntax, please see Chapter 13, [Using Logger Commands](#).

4.3.1. Specifying a UDP Port

By default, the Logger sends and receives data over UDP port 3000.

- To specify a UDP port, use the `(-P | --disPort)` command-line option or the `SetConnectionParameters` command.

Table 4-19: Specifying the UDP port

Interface	Command
GUI	Connection dialog box: Port attribute.
Command line	<code>loggerDIS -P 2345</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>disPort</code> variable is in the default configuration file.

4.3.2. Setting the Exercise ID

- To specify the exercise ID, use the `(-e | --exerciseId)` command-line option or the `SetConnectionParameters` command.

Table 4-20: Specifying the exercise ID

Interface	Command
GUI	Connection dialog box: Exercise ID attribute.
Command line	<code>loggerDIS -e 3</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>exerciseld</code> variable is in the default configuration file.

4.3.3. Setting the Site ID

- To specify the site ID, use the `-s | --siteId` command-line option or the `SetConnectionParameters` command.

Table 4-21: Specifying the site ID

Interface	Command
GUI	Connection dialog box: Site ID attribute.
Command line	<code>loggerDIS -s 3</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>siteId</code> variable is in the default configuration file.

4.3.4. Setting the Application Number

- To specify the application number, use the `-a | --appNumber` command-line option or the `SetConnectionParameters` command.

Table 4-22: Specifying the application number

Interface	Command
GUI	Connection dialog box: Application Number attribute.
Command line	<code>loggerDIS -a 2</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>appNumber</code> variable is in the default configuration file.

4.3.5. Using Asynchronous I/O

You can configure the Logger to use asynchronous I/O. Use of asynchronous I/O can help reduce dropped packets during times of high traffic.

- To specify use of asynchronous I/O, use the `SetConnectionParameters` command.

Table 4-23: Specifying use of asynchronous I/O

Interface	Command
GUI	Connection dialog box: Use Asynchronous IO attribute.
Command line	Use the <code>SetConnectionParameters</code> command.
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>useAsyncIO</code> variable is in the default configuration file.

4.3.6. Specifying a Destination Address for Outgoing and Incoming Data

Normally, the Logger sends outgoing, and receives incoming, data packets using the host machine's default broadcast address. The specified destination address can be a broadcast, multicast, or unicast address.

- To specify a destination address, use the `-A | --destAddrString` command-line option or the `SetConnectionParameters` command.

Table 4-24: Specifying the destination address

Interface	Command
GUI	Connection dialog box: Destination Address attribute.
Command line	<code>loggerDIS -A 205.205.205.205</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>destAddrString</code> variable is in the default configuration file.

4.3.7. Setting the Subnet Mask

When the destination address is not specified ("0.0.0.0"), the subnet mask is used in conjunction with the network interface address to determine the destination broadcast address. If neither is specified (subnet mask is blank), the network interface's default broadcast address is used.

- To specify a subnet mask, use `SetConnectionParameters` command.

By default, the logger will use the subnet mask associated with the IP class.

Table 4-25: Specifying the subnet mask

Interface	Command
GUI	Connection dialog box: Subnet Mask attribute.
Command line	Use the <code>SetConnectionParameters</code> command.
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>subnetMask</code> variable is in the default configuration file.

4.3.8. Setting the Buffer Size for Sending and Receiving PDUs

You can specify the size of the buffers used for sending and receiving DIS packets. A larger receive buffer reduces the likelihood of dropping packets. Buffer size is specified in bytes. A value of -1 means use the system default value.

- To specify send and receive buffer sizes, use the `--receiveBufferSize` and `--sendBufferSize` command-line options or the `SetConnectionParameters` command.

Table 4-26: Specifying the send and receive buffers

Interface	Command
GUI	Connection dialog box: Send Buffer Size and Receive Buffer Size attributes.
Command line	<code>loggerDIS --receiveBufferSize 64000 --sendBufferSize 64000</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>sendBufferSize</code> and <code>recvBufferSize</code> variables are in the default configuration file.

4.3.9. Setting the Multicast Time to Live Value

The multicast time to live value specifies the number of router jumps a message can take.

- To specify the multicast time-to-live value, use the `--mcastTtl` command-line option or the `SetConnectionParameters` command.

Table 4-27: Specifying the multicast time-to-live

Interface	Command
GUI	Connection dialog box: Multicast TTL attribute.
Command line	<code>loggerDIS --mcastTtl 3</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>mcastTtl</code> variable is in the default configuration file.

4.3.10. Setting the Network Interface Address

The network interface address specifies the address to use when a computer has multiple network cards.

- To specify the network interface address, use the `-D` | `--deviceAddrString` command-line option or the `SetConnectionParameters` command.

Table 4-28: Specifying the network interface address

Interface	Command
GUI	Connection dialog box: Network Interface Address.
Command line	<code>loggerDIS --deviceAddrString string</code>
Text interface	Use the <code>SetConnectionParameters</code> command.
<i>lgrConfig.xml</i>	The <code>deviceAddrString</code> variable is in the default configuration file.

4.3.11. Specifying the Heartbeat

The heartbeat value determines how frequently (in seconds) the Logger generates interim PDUs for each entity during slow-speed playback. For more information, please see [“Generating Interim PDUs During Slow-Speed DIS Playback \(Heartbeating\),”](#) on page 3-6.

To set the heartbeat in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the DIS Configuration page ([Figure 4-6](#)).

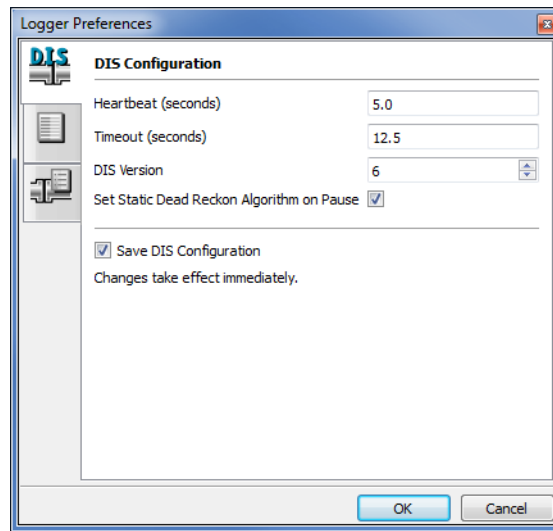


Figure 4-6. Logger Preferences dialog box, DIS Configuration page

3. Type a value in the Heartbeat text box.
- To specify the heartbeat from the command interface, use the `SetHeartbeatRate` command.

Table 4-29: Specifying the heartbeat

Interface	Command
GUI	See procedure in this section.
Command line	<code>loggerDIS -r "SetHeartbeatRate 5.0"</code>
Text interface	<code>SetHeartbeatRate 5.0</code>
<i>lgrConfig.xml</i>	Add <code>SetHeartbeatRate</code> to the configuration file.

4.3.12. Specifying the Timeout

The timeout value determines how long the Logger continues to generate interim PDUs for an entity once that entity's PDUs have stopped occurring in the file. For more information, please see [“Generating Interim PDUs During Slow-Speed DIS Playback \(Heartbeating\),”](#) on page 3-6.

To set the timeout in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
 2. Select the DIS Configuration page ([Figure 4-6](#)).
 3. Type a value in the Timeout text box.
 4. Click OK.
- To specify the DIS timeout from the command interface, use the `SetTimeoutRate` command.

Table 4-30: Specifying the timeout

Interface	Command
Command line	<code>loggerDIS -r "SetTimeoutRate 12.0"</code>
Text interface	<code>SetTimeoutRate 12.0</code>
<i>lgrConfig.xml</i>	Add <code>SetTimeoutRate</code> to the configuration file.

4.3.13. Specifying the DIS Version

The Logger supports DIS versions 4, 5, 6, and 7.

To specify the DIS version:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the DIS Configuration page ([Figure 4-6](#)).
3. Select an option in the DIS Version list.
4. Click OK.

4.3.14. Subscribing to Multicast Groups

In record mode, the Logger can subscribe to multicast groups. When the Logger is subscribed to multicast groups, it also continues to receive data through the machine's broadcast and unicast addresses.

Subscription applies to record mode only.



Multicast subscription is a property of the host machine, not of individual applications. If several applications are running on the same machine using the same port, and those applications subscribe to multicast groups, all the applications will receive packets for all multicast groups to which any application is subscribed. Therefore, if the Logger is using the same port as applications that subscribe to multicast groups, the Logger may receive traffic intended for the other applications.

- To subscribe to multicast groups, use the `-S | --mcastAddresses` command-line option or the `SetConnectionParameters` command.

Table 4-31: Subscribing to multicast groups

Interface	Command
GUI	Connection dialog box: Multicast Addresses attribute.
Command line	<code>loggerDIS -S "225.0.0.118 225.0.0.202"</code>
Text interface	<code>SetConnectionParameters port exerciseId siteId appNumber useAsynchIO sendBufferSize recvBufferSize multicastTtl destAddr deviceAddr "mcastAddresses"</code>
<i>lgrConfig.xml</i>	The <code>mcastAddresses</code> variable is in the default configuration file.



5. Configuring the Logger

This chapter explains how to configure various Logger features and how to edit the *lgrConfig.xml* file.

Configuring the Logger.....	5-2
Configuring Indexing	5-3
Configuring Indexing Using the Command Interfaces	5-4
Configuring Faster Playback	5-5
The SetEntityTracking Command	5-6
The SetFastJumpsWithoutDiff Command.....	5-6
The SetPlaybackReadTimeLimit Command	5-7
Configuring HLA Heartbeating	5-8
Configuring HLA Timeouts.....	5-9
Setting the Dead Reckoning Algorithm on Pause.....	5-9
Configuring Network Read Time	5-10
Using Data Distribution Management.....	5-11
Specifying Subscription Regions.....	5-12
Using Conveyed Regions.....	5-14
Using Simulation Data	5-15
Configuring DDM.....	5-15
Configuring DDM in lgrConfig.xml.....	5-17
Setting the Notification Level	5-18
Specifying the Log File	5-18
Editing lgrConfig.xml.....	5-19
Adding a Command to lgrConfig.xml.....	5-22
Summary of Command-line Options for the Logger.....	5-24

5.1. Configuring the Logger

You can configure the Logger by editing its configuration file, *lgrConfig.xml*, with command-line options, in the text interface, and in the GUI. With the exception of some of the command-line options, all of the ways that you can configure the Logger are essentially alternate ways of implementing Logger commands. For a complete list of command-line options, please see [“Summary of Command-line Options for the Logger,”](#) on page 5-24 or the Quick Reference Card. For descriptions of Logger commands and how to use them, please see Chapter 13, [Using Logger Commands](#) or the Quick Reference Card. For details about the configuration file, please see [“Editing lgrConfig.xml,”](#) on page 5-19.

When you change a configuration setting in the Preferences dialog box, the change gets saved to *lgrConfig.xml* if the Save Configuration check box on that page is selected. If the Save Configuration check box on the Network Settings page is selected, then when you make changes in the Connection dialog box, they get saved to *lgrConfig.xml*.

This chapter describes the configuration options and shows how to implement them in the various interfaces.

5.2. Configuring Indexing

Logger files contain index data (checkpoints) that allow the Logger to quickly and efficiently jump from one time point to another.

When you jump the Logger to a time point, it jumps to the nearest checkpoint before the selected time point and then processes the packets in the file to the specified time point. (The packets are not sent over the network.) Therefore, the efficiency of time jumps is determined by the frequency of checkpoints. Having more frequent checkpoints speeds up time jumps. However, having more checkpoints increases file size and can slow down recording, particularly if the exercise has many entities. So when you specify your checkpoint policy, you must balance your priorities.

When you record, indexing is always enabled. You can set the checkpoint policy (the criteria for how often the Logger adds index data) to one of the following:

- ♦ By Frequency, in packets received (default).
- ♦ By Size, in bytes.
- ♦ By Time.

You can specify that when you export a recording, checkpoints be resampled with the current indexing policy. This allows you to change the indexing policy from that used when the file was recorded.

You can specify that interactions be saved to checkpoints. If interactions are saved, you can specify that the interaction data be sent along with state data when skipping during playback.



If you are currently recording an exercise, changing the preferences does not affect the recording process. It takes effect for the next new file.

To configure indexing in the GUI:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the Indexing Settings page (Figure 5-1). (The Indexing Settings page is the same for HLA and DIS.)

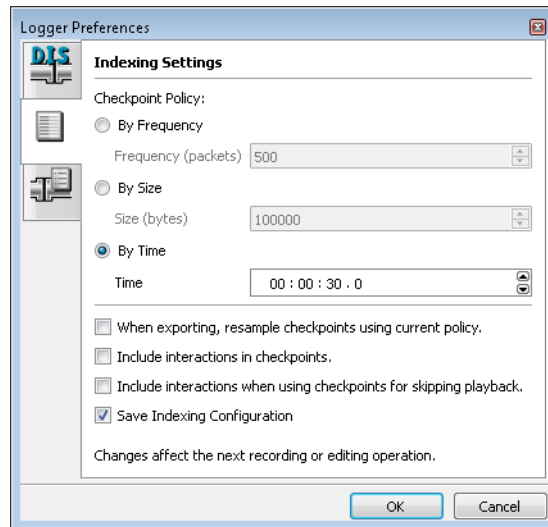


Figure 5-1. Logger Preferences dialog box, Indexing Settings page

3. Select the checkpoint policy.
4. Edit the appropriate value for the checkpoint policy you chose (frequency, size, or time).
5. Optionally select one or more indexing options.
6. Click OK.

5.2.1. Configuring Indexing Using the Command Interfaces

- To configure indexing using a command interface, issue the `SetCheckpointParameters` command.

Table 5-1: Checkpointing command

Interface	Command
Command line	<code>logserver -r "SetCheckpointParameters 0 0 500 0"</code> For details about the parameter values, please see "Recorder Domain Commands," on page 13-10.
Text interface	<code>SetCheckpointParameters 0 0 500 0</code>
<i>lgrConfig.xml</i>	Add <code>SetCheckpointParameters</code> to the configuration file.

5.3. Configuring Faster Playback

There are some processes that you can modify so that the Logger can play back files more quickly at the expense of some data or feature loss. In certain cases, the data loss is acceptable and the improved performance is required because of the density of data or packets in the file. For example, in DIS and some HLA federations, there are entity heartbeats and timeouts. So if you drop some entity state information or miss the removal of an entity, that information can be retrieved by subsequent state updates or entity timeouts.

The loss of data is likely to have a greater effect on HLA federations, which usually do not have an entity heartbeat mechanism. In HLA the object management services inform federates about object discoveries and removals and there are services for requesting missing data. As a result, if the Logger skips data, there is no certainty that future updates will provide the missing data. More importantly, since in HLA the Logger only removes entities when it processes a removal packet, if one is skipped, an object that should have been removed will erroneously continue to persist in the federation.

In both DIS and HLA, all skipped interactions are lost. Since interactions are normally transient events, they can often be missed without great impact. However, if the exchange of interactions contains some state information, for example, transitions in a transaction protocol such as DIS logistics, then a missed interaction may leave the transaction in a bad state.

The Logger uses the following commands to configure fast playback:

- ♦ `SetEntityTracking`
- ♦ `SetFastJumpsWithoutDiff`
- ♦ `SetPlaybackReadTimeLimit`.

In conjunction with fast playback, in HLA the following commands let you configure the Logger to perform entity heartbeats and timeouts and support federations that have these requirements:

- ♦ `SetHeartbeatIntervals`
- ♦ `SetTimeoutIntervals`.

5.3.1. The SetEntityTracking Command

The `SetEntityTracking` command enables or disables the entity tracking process during playback. Entity tracking keeps track of the entities in a Logger file and supplies the list of entities for the entity filter. The entity tracker is periodically updated. Eliminating this overhead speeds up playback. However, entity filtering depends on the entity tracker information and no entities are displayed when entity tracking is disabled. Default: entity tracking is enabled.

The syntax for this command in the text interface is:

```
SetEntityTracking isEnabledValue
```

For example:

```
SetEntityTracking false
```

The syntax for using this command from the command-line is:

```
loggerversion -r "SetEntityTracking isEnabledValue"
```

For example:

```
loggerversion -r "SetEntityTracking false"
```

5.3.2. The SetFastJumpsWithoutDiff Command

The `SetFastJumpsWithoutDiff` command changes how the Logger jumps to a particular time in the file during playback. The default behavior is to first get the nearest checkpoint data. This establishes the initial conditions of the jump (that is, current entities and their state). The Logger then processes the packets from the checkpoint time up to the jump time; however, the output is disabled. The result is that the internal object state is updated, but no data is sent to the network. This two step process establishes the current state of the entities at the selected jump time. When `SetFastJumpsWithoutDiff` is enabled, the Logger jumps immediately from the checkpoint to the jump time without processing intermediate packets. Playback in DIS can handle this fast jump because of entity state updates and entity timeouts. Default: fast jumps are disabled.

The syntax for this command in the text interface is:

```
SetFastJumpsWithoutDiff isEnabledValue
```

For example:

```
SetFastJumpsWithoutDiff true
```

The syntax for using this command from the command-line is:

```
loggerversion -r "SetFastJumpsWithoutDiff isEnabledValue"
```

For example:

```
loggerversion -r "SetFastJumpsWithoutDiff true"
```

5.3.3. The SetPlaybackReadTimeLimit Command

The `SetPlaybackReadTimeLimit` command limits the time the Logger can read packets from a file and drops the remaining packets up to the current time. The Logger records each packet with a relative timestamp from the beginning of the file. When playing back a file, the Logger reads all of the packets that have timestamps less than or equal to the current time. There is no time limit for the length of this read operation. It is possible that the density of a file may be such that the Logger takes too long to read all of the packets. The result is that the Logger begins to lag behind the selected playback speed.

A simple example of how this can happen is if the Logger playback speed is increased such that the Logger cannot read all the packets and maintain the selected speed. This lagging behavior could also happen if a very fast machine records the file and a slow machine is used to play it back or a lightly loaded and then heavily loaded machine. In any case, if the file or playback speed is consistently too dense for the Logger to play back, it falls farther and farther behind.

Setting a read time limit causes the Logger to cut off the read time if needed to maintain the selected speed. However, like the fast jump, the Logger discards data and it should only be employed when dropped data is not a concern or can be recovered by subsequent updates. This trade-off is a bit more drastic in that the Logger may continually drop packets, and while unlikely, the distribution of the packets in the file may be such that some entities may get updated sporadically or not at all. The default configuration disables the read time limit and allows the Logger to take whatever time necessary to read all the packets up to the current time.

When you set the read time limit, it is important to consider the tick rate. The Logger uses the tick rate (really a tick interval) to periodically schedule Logger activity. The Logger starts a read operation (which includes playback of those packets) and when complete, it sleeps a period of time. The sleep allows other processes to run, like the Logger GUI. The sleep interval is established by the tick rate minus the time required to process the read operation. In this way, the Logger performs reads at regular intervals. A candidate value for the read time limit is a value slightly less than the tick rate. In this way, the Logger is assured of completing the read operation before the next scheduled tick time.

This command takes a floating point argument. A value less than or equal to zero disables the behavior. The syntax for this command in the text interface is:

```
SetPlaybackReadTimeLimit timeLimitValue
```

For example:

```
SetPlaybackReadTimeLimit 0.2
```

The syntax for using this command from the command-line is:

```
loggerversion -r "SetPlaybackReadTimeLimit timeLimitValue"
```

For example:

```
loggerversion -r "SetPlaybackReadTimeLimit 0.2"
```

5.3.4. Configuring HLA Heartbeating

HLA entity heartbeating sends a complete update of all the object attributes at a specified interval. The Logger will perform dead-reckoning for entity and aggregate publishers if it is using the internal RPR FOM FOM Mapper or a FOM Mapper library that maps to these publishers. If not, the last known values are used.

HLA heartbeating is configured with the `SetHeartbeatIntervals` command. `SetHeartbeatIntervals` takes as an argument a space-separated set of class heartbeat intervals. Each interval is specified in the form "`className:interval:inheritable`", for instance "ObjectRoot:5.0:true".

The Logger sends a heartbeat for each object as specified by its class interval parameter. If an interval is not specified for an object class, the Logger uses the closest parent class for which there is an interval and the class specified inheritable as true. For example, if all classes are a subclass of ObjectRoot and you want all classes to heartbeat every 5.0 seconds, "ObjectRoot:5.0:true" configures this without requiring every class to be individually listed.

If an object's class specifies a zero interval or the interval is not specified and there is no inheritable interval available, objects of that class do not heartbeat.

The syntax for this command in the text interface is:

```
SetHeartbeatIntervals "className:interval:inheritable ..."
```

You can specify as many classes as needed. Each specification must be separated by a space. For example:

```
SetHeartbeatIntervals "ObjectRoot:5.0:true Object-  
Root.BaseEntity.PhysicalEntity:4.0:true"
```

5.3.5. Configuring HLA Timeouts

You can configure the HLA Logger to time out entities. When recording, if the Logger does not receive an update from a known entity within its timeout interval, it acts as if the entity has been removed. The timeout interval is conventionally the heartbeat interval multiplied by 2.5.

To configure HLA timeouts use the `SetTimeoutIntervals` command. This command takes as an argument a space-separated set of class timeout intervals. Each interval is specified in the form "`className:interval:inheritable`", for instance "`ObjectRoot:12.5:true`".

The Logger removes known objects based on the object's class timeout parameter. If a timeout is not specified for an object class, the Logger uses the closest parent class for which there is a timeout and the class specified inheritable as true. For example, if all classes are a subclass of `ObjectRoot` and you want all classes to timeout after 12.5 seconds, "`ObjectRoot:12.5:true`" specifies this without requiring every class to be individually listed.

If an object's class specifies a zero timeout or the timeout is not specified and there is no inheritable timeout available, objects of that class do not time out.

The syntax for this command in the text interface is:

```
SetTimeoutIntervals "className:interval:inheritable ..."
```

You can specify as many classes as needed. Each specification must be separated by a space. For example:

```
SetTimeoutIntervals "ObjectRoot:12.5:true Object-  
Root.BaseEntity.PhysicalEntity:15.0:true"
```

5.4. Setting the Dead Reckoning Algorithm on Pause

When you pause playback of a Logger file, the Logger sets the Frozen bit for each simulation object. If a receiving application respects the Frozen bit, it stops dead-reckoning the simulation objects. However, if it does not, it may continue to dead-reckon them, which results in incorrect representation of the state of the simulation. To help with this possible problem, when you pause playback, the Logger sets the dead-reckoning algorithm to static, which stops dead-reckoning in the receiving applications.

You can disable this change to the dead-reckoning algorithm. If you make this change, it is up to you to ensure that your applications handle dead-reckoning properly.

To enable or disable changing the dead-reckoning algorithm on pause:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the DIS Configuration or HLA Configuration page, as appropriate.
3. Select or clear the Set Static Dead Reckon Algorithm on Pause check box.
4. Click OK.

5.5. Configuring Network Read Time

You can specify how much time to spend reading messages from the network. Setting a maximum time can prevent the Logger from becoming unresponsive when it is flooded with messages. The options for Network Settings are as follows:

- Read All Messages (HLA) or Read All PDUs (DIS). When selected, during each tick the Logger reads all messages in the queue.
- Max Read Time. Specifies the maximum time to spend reading messages each tick.
- Min Read Time (HLA only). Specifies the minimum amount of time to spend reading messages each tick.

Num PDUs Per Check (DIS only). Specifies the maximum number of PDUs to read before checking to see if the Max Read Time has been reached.



Max Read Time, Min Read Time, and Num PDUs Per Check are not available when Read All Messages is selected.

To configure network read settings:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the Network Settings page (Figure 5-2). (The Network Settings page is similar in DIS and HLA.)

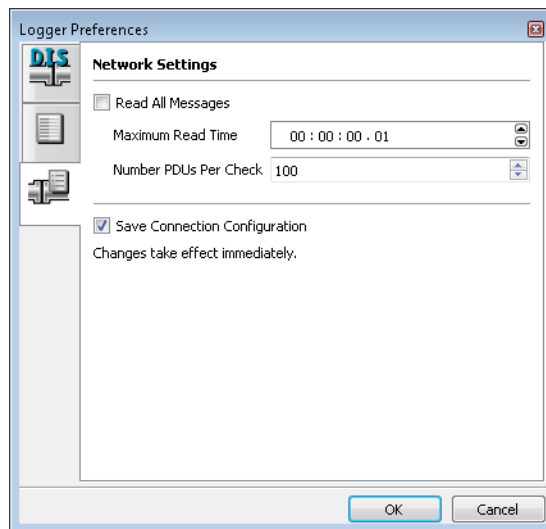


Figure 5-2. Logger Preferences dialog box, Network Settings page

3. To read all messages in each tick, select Read All Messages (for HLA; for DIS, select Read All PDUs.)

4. To specify the maximum amount of time to read messages, clear Read All Messages and type a value for Max Read Time.
5. To specify the minimum amount of time to read messages (HLA only), clear Read All Messages and type a value for Min Read Time.

To specify the maximum number of PDUs to read before checking Max Read Time (DIS only), clear Read All Messages and type a value for Num PDUs Per Check.
6. Click OK.

5.6. Using Data Distribution Management

The Logger can record and play back using HLA Data Distribution Management (DDM). There are two general use cases for configuring DDM with the Logger. The first case uses the HLA DDM conveyed region information to record and play back using DDM regions. The second case supports creating a plug-in to inspect the simulation data to create and update regions. The DDM configuration parameters control which of these approaches is supported and they allow some variation in how they are supported.

In both cases, you can configure the Logger to use either global region subscriptions (*subscribeMode* = **SubscribeRegionsAreGlobal**) or specific region subscriptions (*subscribeMode* = **SubscribeRegionsAreSpecified**). When the Logger subscribes using global regions, it uses an appropriate region with global extents for each class subscription to receive all possible data from the federation. This subscription mode requires no further configuration other than setting the *subscribeMode* parameter in the **SetDdmParameters** command.

You can configure the Logger with specific region subscriptions to restrict the data that the Logger records, or segregate the data among several Loggers, each specifying different region extents. This subscription mode requires a subscription specification file, as described in [“Specifying Subscription Regions,”](#) on page 5-12.

5.6.1. Specifying Subscription Regions

The specification of subscription regions can be read from a configuration file using the `LoadSubscriptionListFile` command. You can include it in the Logger configuration file in the startup section, as follows:

```
<command domain="Simulation" protocol="HLA 13"
  command="LoadSubscriptionListFile">
  <param name="filename" var="subscriptionListFile"></param>
</command>
```

A subscription list file has a list of region descriptors, a list of region sets, and a list of associations between region sets and object and interaction classes. Region descriptors specify the lower and upper bounds of each dimension in the region. The bound values are the normalized values used by the RTI. For the HLA 1.3 specification, the region descriptor includes the routing space name and must specify the bounds for all the dimensions in the routing space.

The following example configuration associates regions with object and interaction classes:

```
<?xml version="1.0" encoding="utf-8"?>
<subscriptionList>
  <regionList name="subRegionList">
    <region name="Region1">
      <space>RoutingSpaceName</space>
      <extent>
        <dimension name="Dim1">
          <lowerBound>0</lowerBound>
          <upperBound>20000</upperBound>
        </dimension>
        <dimension name="Dim2">
          <lowerBound>0</lowerBound>
          <upperBound>20000</upperBound>
        </dimension>
      </extent>
    </region>
    <region name="GlobalRegion">
      <space> RoutingSpaceName </space>
      <extent>
        <dimension name="Dim1">
          <lowerBound>0</lowerBound>
          <upperBound>4294967293</upperBound>
        </dimension>
        <dimension name="Dim2">
          <lowerBound>0</lowerBound>
          <upperBound>4294967293</upperBound>
        </dimension>
      </extent>
    </region>
  </regionList>

  <!-- Each region set contains one or more regions. -->

  <regionSetList name="subRegionSetList">
    <regionSet name="Region1Set">
      <region>Region1</region>
    </regionSet>
```

```

    <regionSet name="GlobalRegionSet">
      <region>GlobalRegion</region>
    </regionSet>
  </regionSetList>

  <!-- In the object and interaction class lists, if a class is not -->
  <!-- associated with a specific region set, it uses the region set -->
  <!-- specified by allObjectClasses or allInteractionClasses. -->
  <!-- If the all*Classes parameter is omitted, and SetDdmParameters -->
  <!-- subscribeUnspecifiedAttributes is disabled, only those classes -->
  <!-- that are explicitly associated with a region set are subscribed. -->

  <classList name="objectClassList">
    <allObjectClasses regionSet="GlobalRegionSet"/>
    <objectClass regionSet="RegionSet1">ObjectRoot.Class1</objectClass>
    <objectClass regionSet="RegionSet1"> ObjectRoot.Class2</objectClass>
  </classList>

  <classList name="interactionClassList">
    <allInteractionClasses regionSet="GlobalRegionSet"/>
    <interactionClass regionSet="RegionSet1">InteractionRoot.Class1
  </interactionClass>
    <interactionClass regionSet="RegionSet1">InteractionRoot.Class2
  </interactionClass>
  </classList>
</subscriptionList>

```

If individual attributes of a class must be associated with specific regions, then the regions can specify those attributes, as follows:

```

<region name="Region1">
  <space>RoutingSpaceName</space>
  <extent>
    <dimension name="Dim1">
      <lowerBound>0</lowerBound>
      <upperBound>20000</upperBound>
    </dimension>
    <dimension name="Dim2">
      <lowerBound>0</lowerBound>
      <upperBound>20000</upperBound>
    </dimension>
  </extent>
  <attribute class="ObjectRoot.Class1" name="Attribute1"/>
  <attribute class="ObjectRoot.Class1" name=" Attribute2"/>
  <attribute class="ObjectRoot.Class1" name=" Attribute3"/>
</region>

```

You must define a separate region for each set of class attributes for a given class and the class must still be associated with the region in the object class list. The class is subscribed with the specified list of attributes and region. If any attributes of the class are left unspecified, and the **subscribeUnspecifiedAttributes** parameter is disabled, they are not subscribed.

Loading the Subscription List Using the API

The subscription list file described in the previous section can be loaded through the API as follows:

```
void loadSubscriptionFile(DtLogger* logger, const DtString& fileName)
{
    DtSubscriptionListFile* sublist = DtSubscriptionListFile::create();
    sublist->readFromFile(fileName);
    logger->simManager()->setSubscriptionList(*sublist);
    delete sublist;
}
```

Alternatively, the subscription list can be constructed incrementally. For details, please contact support@mak.com.

5.6.2. Using Conveyed Regions

The HLA DDM service supports the concept of conveyed regions. Conveyed regions provide the receiving federate the region extents associated with the reflected attributes or received interactions at the time the data was sent. By recording these conveyed region extents, the Logger can reproduce the regions during playback.

You can configure the Logger to record the conveyed region information. This region information can then be used during playback to create regions, update regions, and switch object attributes between regions. The conveyed regions provided by the RTI during Federate Ambassador callbacks specify the region extents that were active when the object instance attributes were updated or the interaction was sent. The Logger records these region extents with the received data. When playing back a file with region information, the Logger can send the data using the recorded region extents. Because the region information was provided by the RTI and recorded by the Logger, it is not necessary to inspect the data in order to create and update regions. The Logger replicates the same region extents as when the data was sent by the original federate.



To record conveyed region information, the RTI must be configured to deliver the conveyed region information in the Federate Ambassador callbacks. See the appropriate RTI documentation for enabling this HLA feature.

5.6.3. Using Simulation Data

The Logger plug-in API provides class methods used in the playback process that can be overridden to inspect the outgoing data to create and update regions to support DDM. DDM regions are used to send interactions and they are associated with object instance attributes. The interaction data can be inspected to select a region and then the region extents can be modified prior to sending the interaction. The object data can be inspected when the object publisher is created in order to select the regions used to register the object. As the object is updated, the region extents can be modified or the attributes can be associated with different regions.

5.6.4. Configuring DDM

The DDM settings are set using the `SetDdmParameters` command in the Logger configuration file, text interface, or on the command line. The syntax for this command in the text interface is:

```
SetDdmParameters enableDdm playbackMode subscribeMode
playbackDynamicAssociation playbackDynamicExtents
allowPlaybackWithoutRegions recordConveyedRegions
requestUpdates subscribeUnspecifiedAttributes
```

For example:

```
SetDdmParameters true PlaybackRegionsUseConveyedExtents
SubscribeRegionsAreGlobal true false true true true true
```

The parameters for the `SetDdmParameters` command are as follows:

- ♦ *enableDdm*. Enable record and playback using DDM. Default: false.
- ♦ *playbackMode*. Supported modes are:
 - **PlaybackRegionsUseConveyedExtents**. During playback, regions are created and updated using recorded region data.
 - **PlaybackRegionsUseData**. Regions are created and updated by inspecting simulation data. Requires a plug-in.
 - **PlaybackRegionNotEnabled** (default). Updates and interactions are sent without regions.
- ♦ *subscribeMode*. Supported modes:
 - **SubscribeRegionsAreGlobal**. DDM subscriptions are made with global regions with extents of global min and max.
 - **SubscribeRegionsAreSpecified**. The regions are specified through subscription configuration files. (For details, please see the `LoadSubscriptionListFile` command.) The subscription region configuration can also be specified programmatically through the Logger plug-in API.
 - **SubscribeRegionNotEnabled** (default). Subscriptions are performed without regions.

- ♦ *playbackDynamicAssociation*. During playback, the association of regions with object instance attributes can be fixed or dynamic. If it is known that region associations with attribute instances never change, the overhead of checking for changes can be eliminated. This parameter should be disabled if possible. Default: false.
- ♦ *playbackDynamicExtents*. During playback, the region extents can be fixed or dynamic. The conventional notion of DDM regions regards them as dynamically changing extents to reflect the state of simulation data (for example, surrounding an entity's position). An alternative approach is for the regions to be fixed and the object attribute instances move from region to region. If it is known that region extents never change, the overhead of checking for changes can be eliminated. This parameter should be disabled if possible. Default: false.
- ♦ *allowPlaybackWithoutRegions*. During playback, it may happen that a region association cannot be established. In these cases, the Logger sends the data without region association. Disable this parameter if the federation agreement requires that all data must be transmitted with associated regions. Default: true.
- ♦ *recordConveyedRegions*. Enable this parameter to record conveyed regions. Default: True.
- ♦ *requestUpdates*. Requesting updates can result in spikes in network traffic, especially since it is often performed by late joining federates when the exercise is already populated with many objects. In some DDM federations, object heartbeats are used instead of update requests. Disable this parameter to prevent the Logger from making requests for updates.



The Logger does not use region information to perform update requests. The default value is True.

- ♦ *subscribeUnspecifiedAttributes*. The Logger may not be able to determine a region association to apply to an interaction class or object class attribute subscription. This situation may occur if the interaction or class does not specify a routing space or dimension in the FOM. It can also happen if the *subscribeMode* is configured as **SubscribeRegionsAreSpecified**, but some classes or attributes are left unspecified. By default, the Logger subscribes to these interaction classes and object class attributes without using a region association (in effect a global subscription). Disable this parameter if the Logger should only subscribe when a region association can be used. Default: true.

5.6.5. Configuring DDM in *lgrConfig.xml*

The following is an example of configuring DDM in the Logger configuration file:

```
<defaults>
*
*
*
  <var name="playbackModeString" type="string"
    value="PlaybackRegionsUseConveyedExtents" />
  <!-- <var name="playbackModeString" type="string"
    value="PlaybackRegionUsesData" /> -->
  <var name="subscribeModeString" type="string"
    value="SubscribeRegionsAreGlobal" />
  <!-- <var name="subscribeModeString" type="string"
    value="SubscribeRegionsAreSpecified" /> -->
  <var name="True" type="bool" value="true" />
  <var name="False" type="bool" value="false" />
</defaults>

<commands name="startup">
*
*
*
  <command domain="Simulation" protocol="HLA 13"
    command="SetDdmParameters">
    <param name="enableDdm" var="False"></param>
    <param name="playbackMode" var="playbackModeString"></param>
    <param name="subscribeMode" var="subscribeModeString"></param>
    <param name="playbackDynamicAssociation" var="True"></param>
    <param name="playbackDynamicExtents" var="False"></param>
    <param name="allowPlaybackWithouRegions" var="True"></param>
    <param name="recordConveyedRegions" var="True"></param>
    <param name="requestUpdates" var="True"></param>
    <param name="subscribeUnspecifiedAttributes" var="True"></param>
  </command>
</commands>
```

5.7. Setting the Notification Level

The notification level determines the verbosity of informational messages that the Logger generates. The range is 0-4, with 0 being least verbose. Some informational messages are generated by the Logger application and some are generated by the VR-Link libraries it is built against. You can set the notification level for each message source independently.

- To configure the Logger notification level using a command interface, issue the `-m` (for Logger messages) and `-n` (for VR-Link messages) command-line options or the `SetNotifyLevel` command.

Table 5-2: Notification level commands

Interface	Command
Command line	<code>loggerversion -m 2 -n 2</code> or <code>loggerversion -r "SetNotifyLevel 2 2"</code>
Text interface	<code>SetNotifyLevel 2 2</code>
<i>lgrConfig.xml</i>	The <code>SetNotifyLevel</code> command is in the default configuration file.

5.7.1. Specifying the Log File

The Logger sends notification messages to a log file. The default log file is *logger.log*. If you assign a log file a name with the extension *.lgr*, the Logger adds a *.log* extension to ensure that the log file is not confused with a Logger file, or worse, overwrites a file.

- To configure the log file, issue the `-l` command-line option.

Table 5-3: Specifying the log file

Interface	Command
Command line	<code>loggerversion -l mylogfile.log</code>
Text interface	None.
<i>lgrConfig.xml</i>	The <code>logFileName</code> variable is in the default configuration file.

5.8. Editing *lgrConfig.xml*

The *lgrConfig.xml* file contains commands that get run when you start the Logger. The default configuration file starts the Logger with initialization commands to load plug-ins and connect to the DIS or HLA network.

The *.bin* directory has a version of *lgrConfig.xml* for each version of the Logger. The file name includes the protocol, for example *lgrConfigDIS.xml* and *lgrConfigHLA1516.xml*. Default versions of *lgrConfig.xml* are in *.data/backups*. You can also use a default version by specifying it with the `--configFile` command-line option.

The file is written using XML syntax. It has essentially two sections – variable definitions and commands. Variable definitions are specified in the `<defaults>` block. Commands are specified with a series of `<command>` entries within one of two `<commands>` blocks. You can use any Logger command in a configuration file. (For a list of commands, please see “[Command Descriptions](#),” on page 13-4.)



Some command line options have a matching variable name in the defaults section of the configuration file. If you change or remove the name of one of the variables in the default configuration file, it will break the command line option associated with it.

Additionally, if you add a new variable and give it a name that matches an existing command line argument, you will get unexpected behavior. Being careful about how you name new variables should eliminate any risk of this error occurring.

The following is a heavily edited version of a configuration file for HLA that shows the main components:

```
<lgrConfig>
  <!-- Variable formatting -->
  <defaults>
    <var name="messageLevel" type="int" value="2"/>
    <!-- HLA Settings -->
    <var name="execName" type="string" value="MAK-RPR-2.0"/>
    <var name="federateName" type="string" value="MAK Logger"/>
    <var name="fedFileName" type="string" value="MAK-RPR2-1-1.fed"/>
    <var name="fomMapperLib" type="string" value=""/>
    <var name="fomMapperData" type="string" value=""/>
    <var name="rprFomVersion" type="double" value="2.0"/>
    <var name="rprFomRevision" type="int" value="1"/>
    <var name="ignoreAdvisoriesValue" type="bool" value="false"/>
    <var name="useEmptyFomMapperValue" type="bool" value="false"/>
  </defaults>

  <!-- Initialize commands executed after the command line is parsed -->
  <!-- Parameter variables must be defined in the defaults group -->
  <commands name="init">
    <command domain="System" protocol="None" command="SetNotifyLevel">
      <param name="lgrLevel" var="messageLevel"></param>
    </command>
  </commands>

  <commands name="startup">
    <command domain="Simulation" protocol="HLA 13"
      command="SetConnectionParameters">
      <param name="federateName" var="federateName"></param>
      <param name="federationName" var="execName"></param>
      <param name="fedFile" var="fedFileName"></param>
      <param name="fomVersion" var="rprFomVersion"></param>
      <param name="fomRevision" var="rprFomRevision"></param>
      <param name="fomMapper" var="fomMapperLib"></param>
      <param name="fomMapperData" var="fomMapperData"></param>
      <param name="ignoreAdvisories" var="ignoreAdvisoriesValue">
        </param>
      <param name="useEmptyFomMapper" var="useEmptyFomMapperValue">
        </param>
      </command>
    <command domain="Simulation" protocol="None" command="Connect"/>
  </commands>
</lgrConfig>
```

Command Syntax

Commands that do not take arguments have the format:

```
<command domain="domain" protocol="protocol" command="command" />
```

where:

- ♦ *domain* is System, Player, Recorder, Editor, or Simulation. Please see “[Command Descriptions](#),” on page 13-4 for a list of commands by domain.
- ♦ *protocol* is None, DIS, HLA 13, HLA 1516, or HLA 1516e.
- ♦ *command* is the command you want to run, from the list of supported commands in Chapter 13, [Using Logger Commands](#).

The following is an example of a command that does not take arguments:

```
<command domain="Simulation" protocol="None" command="Connect" />
```

Commands that take arguments have the format:

```
<command domain="domain" protocol="protocol" command="command">
  <param name="name" var="variableName"></param>
</command>
```

where:

- ♦ *name* is the name of a command argument as listed in the command descriptions in “[Command Descriptions](#),” on page 13-4.
- ♦ *variableName* is the user-defined name of a variable that is to be defined in the <defaults> section of the configuration file.

The following is an example of a command that takes arguments:

```
<command domain="System" protocol="None" command="SetNotifyLevel">
  <param name="lgrLevel" var="messageLevel"></param>
</command>
```

Variable Syntax

Variable entries take the format:

```
<var name="variableName" type="type" value="value" />
```

where:

- ♦ *variableName* is a variable that is specified by a command in the commands section of the configuration file.
- ♦ *type* is the argument type for the variable, as defined for each command argument in “[Command Descriptions](#),” on page 13-4.
- ♦ *value* is the value you want to assign to the variable.

For example:

```
<var name="messageLevel" type="int" value="2" />
```

5.8.1. Adding a Command to *lgrConfig.xml*

When you add a command to the configuration file, you will usually add it to the startup commands section. The only time you would add a command to the init section is if you are adding a command to load a plug-in.

Suppose that you want to add the `NewFile` command to *lgrConfig.xml*. This command creates a new file for recording. If you add it to the configuration file, every time you start the Logger, it will create a new file with the specified name.

1. Add a new command in the `<commands name="startup">` section, similar to the following:

```
<commands name="startup">
  <command domain="Recorder" protocol="None" command="NewFile">
    <param name="filename" var="newfile"></param>
  </command>
</commands>
```

In the command definition, the domain is `Recorder`, which you know because `NewFile` is listed with the commands in the `Recorder` domain in “[Command Descriptions](#),” on page 13-4. Protocol is `None`, because the command is protocol independent. The command is `NewFile`.

The command definition shows that `NewFile` takes an argument, `filename`, of type `string`. Therefore, add a parameter definition to the command. The name is `filename`. This must be exactly the same spelling as in the definition. The variable name is a user-defined name that will be defined in the variable definition.

2. Add a variable definition for the variable `newfile`, similar to the following:

```
<var name="newfile" type="string" value="newfilename.lgr"/>
```

The name, `newfile`, matches the variable name in the command entry. The type is the type listed in the command definition. The value is whatever value you want to assign, in this case, the name you want to give to the Logger file that will be created on startup.

Adding a String List to lgrConfig.xml

Some commands require specification of multiple strings, for example, the list of common entity types in the configuration file for DIS. As with other commands, you specify a variable to take the data. Then you provide the strings as a list of listitems. For example, here is how you could specify a subscription list for HLA:

1. In the command section of the configuration file, add an entry for the `SetSubscriptionLists` command, as follows:

```
<!-- To Disable auto-connection remove this command -->
  <command domain="Simulation" protocol="None" command="Connect"/>
</commands>

<command domain="Simulation" protocol="HLA 13"
  command="SetSubscriptionLists">
  <param name="objectClasses" var="objectSubscriptionList"></param>
  <param name="interactionClasses" var="interactionSubscriptionList">
    </param>
</command>
```

Since the `SetSubscriptionLists` command takes two arguments, we define two parameters for it.

i

We include the `Connect` command, which is in the default configuration file, because the Logger must be connected before you can issue a `SetSubscriptionLists` command. When you use Logger commands, you must take into account any prerequisites that they have.

2. Add a variable definition for each of the parameters defined in the command. Within each variable definition, include a listitem for each string you want to include in the list.

```
<var name="objectSubscriptionList" type="list" listtype="string">
  <listitem>ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft
  </listitem>
  <listitem>ObjectRoot.BaseEntity.PhysicalEntity.Radio</listitem>
</var>

<var name="InteractionSubscriptionList" type="list" listtype="string">
  <listitem>InteractionRoot.WeaponFire</listitem>
</var>
```

i

The subscription list created by this example is identical to the one that would be created by the example command in [“Using Logger Commands on the Command Line,”](#) on page 13-2 and [“Using Logger Commands in the Text Interface,”](#) on page 13-3.

5.9. Summary of Command-line Options for the Logger

Table 5-4 lists command-line options. Command-line options override the comparable settings in the *lgrConfig.xml* file.

Table 5-4: Command-line options (Sheet 1 of 3)

Option	Description
<code>--allowAllRemoteControl mode</code>	Specifies the response to remote control PDUs. 1 = Allow 2 = Do not allow.
<code>{-c --showConsole} mode</code>	Enables (1) or disables (0) display of a console window.
<code>--configFile filename</code>	Specifies the configuration file to load.
<code>--defaultTapeDir directory</code>	Specifies the default directory for Logger files.
<code>filename</code>	Specifies a Logger file to load. This option uses an implicit <code>-r OpenFile</code> . If a filename starts with a hyphen (-), you must use the <code>-r OpenFile</code> syntax.
<code>--filterEntityTypes entity_type</code>	Specifies an entity type to be added to the Filter by Entity Type dialog box. Filtering will automatically be enabled. Can be used multiple times.
<code>{-- --ignore_rest}</code>	Ignore all arguments after this one.
<code>{-h --help}</code>	Displays a brief summary of Logger command-line arguments.
<code>{-l --logFileName} filename</code>	Specifies a file to send notification messages to. Default: <i>logger.log</i> .
<code>--loggerTickRate rate</code>	The interval for the Logger tick rate, in milliseconds. For more information, please see “The SetPlayback-ReadTimeLimit Command,” on page 5-7
<code>{-m --messageLevel} level</code>	Specifies the notification level for Logger messages for the log file. Range: 0-4.
<code>{-n --notifyLevel} level</code>	Specifies the notification level for VR-Link messages for the log file. Range: 0-4.
<code>--remoteControlId ID</code>	Specifies the Logger's remote control ID. By default the value is assigned the Logger's site ID. This value is used if <code>SetRemoteControlSettings</code> <code>allowAllRemoteControl</code> is disabled.
<code>{-r --runCommand} command</code>	Specifies a Logger command to run. You can specify <code>-r</code> multiple times. For more information, please see “Using Logger Commands on the Command Line,” on page 13-2.
<code>{-v --version}</code>	Displays version information and exits.

Table 5-4: Command-line options (Sheet 2 of 3)

Option	Description
HLA Only	
<code>--disableTMTesting mode</code>	Disable time management for testing.
<code>--enableTrafficAnalyzer mode</code>	Enable (1) or disable (0) the HLA Traffic Analyzer.
<code>{-f --fomMapperLib} libname</code>	Specifies the FOM Mapper to use if you are not using the default FOM. For details about how the Logger chooses the FOM Mapper, please see “Specifying a FOM Mapper,” on page 4-13.
<code>--federateType type</code>	Specifies the federate type for the Logger federate.
<code>--filterObjectNames names</code>	Filter the listed object names. Accepted multiple times.
<code>--fomMapperData data</code>	Specifies FOM Mapper initialization data, if required.
<code>--fomModules “module1 module2 ... moduleN”</code>	Specifies one or more FOM modules for the exercise. (HLA Evolved only)
<code>{-F --fedFileName} fedfile</code>	Specifies the FED file for the exercise.
<code>--ignoreAdvisories mode</code>	Specifies whether (1) or not (0) the Logger ignores advisory messages.
<code>--maxDrainInputTime time</code>	Specifies the maximum time to drain input.
<code>--MIM MIM_file</code>	Specifies the MIM (MOM and Initialization Module) for the exercise. (HLA Evolved only)
<code>{-N --federateName} federate_name</code>	Specifies the federate name for the Logger federate. (HLA Evolved only)
<code>--rprFomVersion version</code>	Specifies the version of the RPR FOM to use.
<code>--rprFomRevision revision</code>	Specifies the FOM revision to use.
<code>--timeConstrainingRecording mode</code>	Use time constrained for recording.
<code>--timeRegulatingPlayback mode</code>	Use time regulating for playback
<code>--useEmptyFomMapper mode</code>	Specifies whether (1) or not (0) the Logger uses an empty FOM Mapper. For details about how the Logger chooses the FOM Mapper, please see “Specifying a FOM Mapper,” on page 4-13.
<code>{-x --execName} ExName</code>	Specifies the federation execution name. Default: VR-Link.
DIS Only	
<code>{-A --destAddrString} address</code>	Specifies the default IP address to which the Logger sends PDUs. If this option is not used, the Logger uses the computer’s broadcast address. Please see “Specifying a Destination Address for Outgoing and Incoming Data,” on page 4-26.

Table 5-4: Command-line options (Sheet 3 of 3)

Option	Description
<code>{-a --appNumber} appnumber</code>	Specifies the application number.
<code>{-D --deviceAddrString} address</code>	Specifies the network interface address if your computer has more than one network card.
<code>--disVersion num</code>	Specifies the DIS protocol version for PDUs sent onto the network. <i>num</i> can be 4 (DIS 2.0.4), 5 (IEEE 1278.1), or 6 (DIS 2.1.4, IEEE 1278.1a):
<code>{-e --exerciseId} ID</code>	Specifies the exercise ID.
<code>--filterEntityIds IDs</code>	Filter the specified entity IDs. Acceptable multiple times.
<code>--mcastTtl ttl</code>	Specifies the multicast time-to-live.
<code>{-P --disPort} port</code>	Specifies which DIS port (UDP port if -A not specified, otherwise, TCP) the Logger uses to send or receive data. Default: 3000.
<code>--recvBufferSize size</code>	Specifies the receive buffer size.
<code>{-S --mcastAddresses} addresses</code>	Subscribes the Logger to the specified multicast address. To subscribe to several multicast addresses, use this option more than once on the command line.
<code>--sendBufferSize size</code>	Specifies the send buffer size.
<code>{-s --siteId} ID</code>	Specifies the site ID.



6. Starting the Logger

This chapter explains how to use the Logger's core features from the GUI.

Starting the Logger	6-2
Starting the Logger on Linux.....	6-2
Starting the Logger in Windows.....	6-2
Running the Logger without a GUI	6-2
The Logger Window.....	6-3
The Timeline	6-4
Hiding Plug-in Views.....	6-5
Docking and Undocking GUI Plug-in Views and Toolbars	6-6
Connecting to the Simulation Network.....	6-7
Disconnecting from the Simulation Network.....	6-7
Opening Logger Files.....	6-7
Upgrading a Logger File's Format	6-8
Opening Incomplete Files	6-9
Closing a Logger File.....	6-9
Listening to the Network.....	6-9
Viewing and Editing Logger File Information.....	6-10
Displaying Information about a Logger File	6-10
Editing the Logger File Description.....	6-11

6.1. Starting the Logger

The Logger has separate executables for each protocol supported (please see [Table 2-1](#) for a description of the directory structure). You can use command-line options to override default configuration parameters and otherwise specify the Logger's initial settings. "Summary of Command-line Options for the Logger," on page 5-24, lists command-line options for the Logger.



The Logger has separate executables for each version provided: loggerDIS, loggerHLA13, loggerHLA1516, and loggerHLA1516e. In sample command lines, we use the generic form `loggerversion`.

6.1.1. Starting the Logger on Linux

To start the Logger:

1. Open a Linux shell.
2. Change to the *bin* directory.
3. Enter:

```
loggerversion [filename options]
```

where *version* is DIS, HLA13, HLA1516, or HLA1516e.

6.1.2. Starting the Logger in Windows

Start the Logger in any of the following ways:

- On the Start Menu, choose **Programs** → **MÄK Technologies** → **Logger 5.4** → **Logger protocol**, where *protocol* is DIS, HLA 1.3, HLA 1516, or HLA Evolved.
- In a command window, change to the *bin* directory, and type:

```
loggerversion [filename options]
```
- Create a shortcut on the Windows desktop and specify the appropriate command line in the Properties dialog box.
- In the Windows Explorer, double-click a Logger file.

6.1.3. Running the Logger without a GUI

You can run the Logger in console mode with or without the GUI. For details about running in console mode, please see "Using Logger Commands in the Text Interface," on page 13-3. To disable the GUI, in the configuration file, comment out the line that installs the protocolGuiLibrary plug-in, as follows:

```
<!-- <command domain="System" protocol="None" command="LoadPlugin">  
  <param name="pluginName" var="protocolGuiLibrary"/>  
</command> -->
```

6.2. The Logger Window

The Logger window has a set of control buttons, a timeline, and a status bar (Figure 6-1). Data supplied by plug-ins, such as the packet views and annotations, is displayed between the toolbar controls and the timeline. You can enable and disable the display of plug-in data.

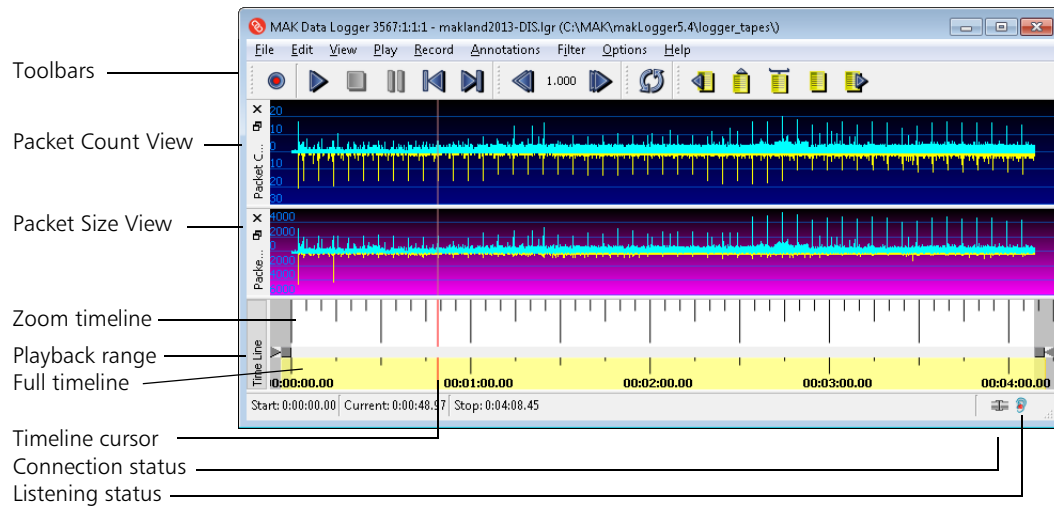


Figure 6-1. Logger window with a Logger file loaded

When you close the Logger, it remembers the size of the window and which views are displayed and restores that configuration the next time you run it.

The Logger window toolbars have the following buttons:

-  Record.
-  Play.
-  Stop.
-  Pause.
-  Jump to Start. Move the timepoint (playback cursor) to the start of the playback range.
-  Jump to End. Move the timepoint to the end of the playback range.
-  Decrease the playback speed (by 50%).
-  Current speed.
-  Increase the playback speed (double the speed).
-  Set the end of playback action (stop or wrap).
-  Jump to the previous annotation.
-  Insert a point annotation.
-  Insert a range annotation.
-  Extend the selected annotation.
-  Jump to the next annotation.

6.2.1. The Timeline

The timeline is actually two timelines (Figure 6-2). During playback, the upper timeline, or zoom timeline, lets you zoom in and out on the recording for a finely grained analysis of what happens at various time points. The bottom half of the timeline, the full timeline, always displays a timeline for the entire length of the recording. The yellow area shows the portion of the recording that is visible in the zoom timeline.

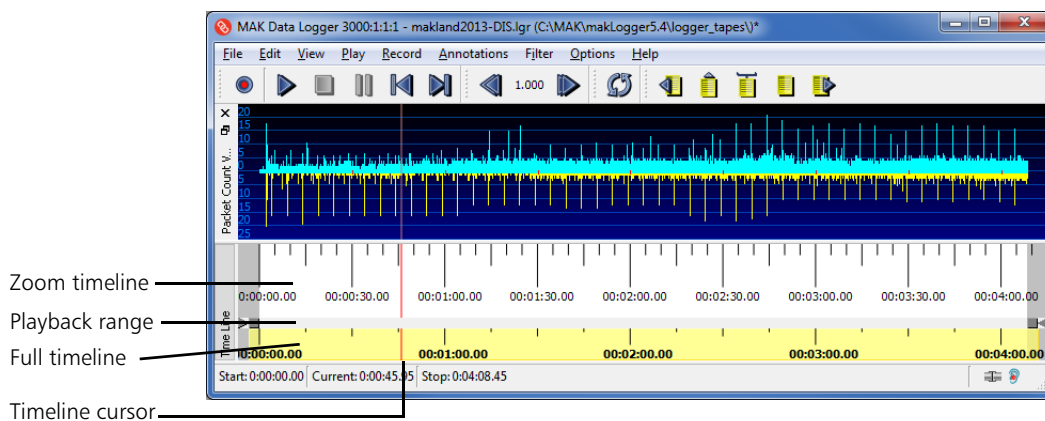


Figure 6-2. Timeline

During recording, the zoom timeline always shows the most recent 30 seconds of the recording. (You can change the amount of time shown with the `SetFollowingInterval` command.) The full timeline shows the entire length of the recording.

The full timeline always shows the location of the timeline cursor. The zoom timeline shows the cursor if it is within the current zoom range.

For more information about the timeline, please see [“Playing a Segment of a File,”](#) on page 7-8, [“Zooming the Timeline,”](#) on page 7-13, and [“Moving the Timepoint to a Specific Time,”](#) on page 7-15.

6.2.2. Hiding Plug-in Views

To hide a plug-in view, use one of the following methods:

- ♦ Select the appropriate option on the View menu.
- ♦ Click the hide button for the plug-in view that you want to hide ([Figure 6-3](#)).

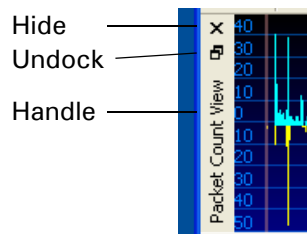


Figure 6-3. Plug-in window controls

6.2.3. Docking and Undocking GUI Plug-in Views and Toolbars

You can undock and dock plug-in views and toolbars from the Logger window and place them anywhere on the desktop. You can dock the toolbars to the window at the top and sides and can reorganize the order in which they are placed.



Floating windows are supported only in the Windows version of the Logger.

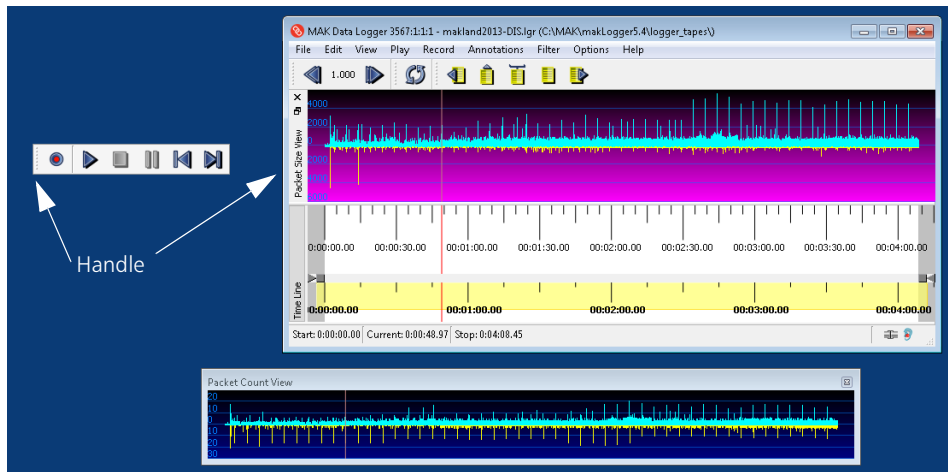


Figure 6-4. Undocked GUI components

- To undock a plug-in view or toolbar, click and hold its handle and drag it off the window. You can also undock a data view by clicking the undock button (Figure 6-3).
- To dock a toolbar, drag it to the Logger window where you want to dock it. You can dock toolbars to the top and sides of the window.
- To dock a plug-in view, drag it to the window or double-click its title bar. You can dock plug-in views in the space above the timeline.



If the Logger window is resized such that there is not enough room above the timeline for a plug-in view, you might not be able to dock a previously undocked view. If this happens, double-click the title bar of the undocked view.

6.3. Connecting to the Simulation Network

When you start the Logger, it automatically connects to the simulation network using the stored connection parameters. If you disconnect from the network, you can connect again as described in this section and as described in [“Editing a Connection,”](#) on page 4-4.

- To connect to the network, choose **File** → **Connect**, or double-click the connection status icon on the status bar (.

6.3.1. Disconnecting from the Simulation Network

When you disconnect from an exercise, you can no longer send data to it or record data from it.



You cannot disconnect from an exercise while you are recording or playing a file.

- To disconnect from an exercise from the GUI, choose **File** → **Disconnect**, or double-click the connection status icon on the status bar (.

6.4. Opening Logger Files

If you want to play a file or edit it, you must open the file. When you open a file, its filename is displayed in the Title Bar. If the Packet Count and Packet Size views are enabled, histograms are calculated and displayed. (For details about the packet views, please see [“Viewing Packet Histograms,”](#) on page 7-16.)

To open a Logger file:

1. Choose **File** → **Open**. The Open File dialog box opens.
2. Select the file. You must select a file that was created with the same protocol as the version of the Logger you are running (DIS, HLA 13, HLA 1516, or HLA Evolved). If you select a file that is for the wrong protocol, the Open button does not get activated. Information about the file is displayed in the Information panel ([Figure 6-5](#)). It includes the protocol used to record the file. (If the Information panel is not displayed, click the Information Panel icon (.

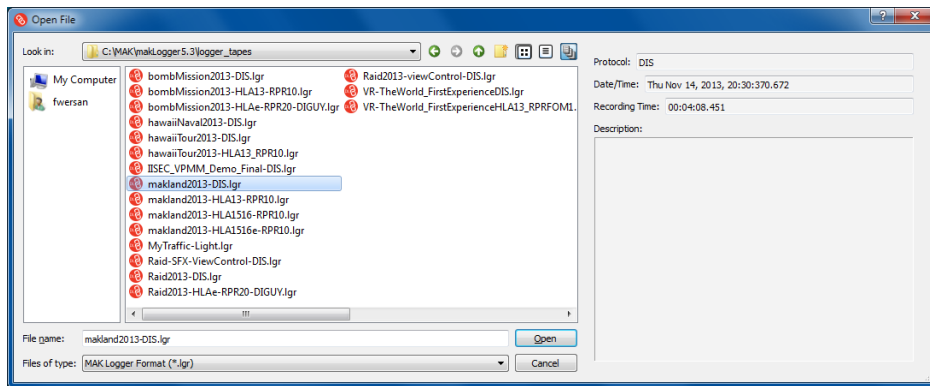


Figure 6-5. Open File dialog box

3. Click Open. If there is an annotations file with the same name as the Logger file, you are prompted to open it. If Play On Load is enabled, the Logger starts playing the file immediately.

i

- You can also load a file by double-clicking its icon in a folder or by dragging it onto the Logger window.
 - If you load a file that was created using a Logger format prior to Logger 5.4, you are prompted to upgrade to the current format. If you decline to do so, some Logger features may not be supported. You can upgrade manually at a later point. For details, please see [“Upgrading a Logger File’s Format,”](#) on page 6-8.
 - If you load a file that was made with a previous version of the Logger and which is not indexed, the Logger does not index the file. Therefore, time jumps may take longer than they would for indexed files.
-

6.4.1. Upgrading a Logger File’s Format

When you load a Logger file that was created with a previous format, the Logger prompts you to upgrade. If you click OK, it upgrades to the latest file format and opens the file. You can also upgrade manually.

To upgrade a file to the latest format:

1. Open the file.
2. Choose **Edit** → **Upgrade File Format**.

6.4.2. Opening Incomplete Files

When the Logger finishes recording a file, it writes a table of all entities and a checkpoint index table to the end of the file. It uses this data to load and play back the file. If recording terminates abnormally, for example due to a crash or running out of disk space, this data is not written. The Logger can repair and open such files.

When the Logger loads a file that is missing the end-of-file data, it prompts you to repair the file. Your options for repairing files are as follows:

- ♦ Repair the file. If you choose this option, the Logger uses the data in the incomplete file and the current checkpointing policy to create a new, complete file, which it then loads. It saves the repaired file to the same location as the incomplete file. The new file is named *original_filenameRepaired.lgr* (for example, *../logger_tapes/demo.lgr* would become *../logger_tapes/demoRepaired.lgr*). The original file is unchanged.
- ♦ Do not repair the file. If you choose this option, then the partial file is loaded. It has the original checkpoints if they are in the file; however the checkpoints may be unreliable. Also, when you load an incomplete file, it does not load the entity tracking information. The entity tracking table provides the entity information for filtering by entity name (entity ID in DIS). Therefore, filter by entity name will have an empty list.

The partial file cannot be saved or saved to another filename. However, it can be exported, which will create a repaired file. Also, editing the file will repair it and allow it to be saved using the original name. The entity tracking information will be restored to the loaded, edited copy.

6.4.3. Closing a Logger File

- To close a Logger file, choose **File** → **Close**.

6.5. Listening to the Network

The Logger listens to the network when it is recording. It can also listen to the network when it is idle; that is, it is not recording or playing a file. This feature allows it to discover objects before it starts recording, which improves performance when recording starts. If you are using HLA, the HLA Traffic Analyzer can view data.

The ability to listen to the network is enabled by default. The listening status is shown by the ear icon on the Status bar (Figure 6-1). You can disable this feature with the `SetListenWhenIdle` command.

```
<var name="listenWhenIdle" type="bool" value="true"/>

<command domain="Simulation" protocol="None"
  command="SetListenWhenIdle">
  <param name="listenWhenIdle" var="listenWhenIdle"></param>
</command>
```

6.6. Viewing and Editing Logger File Information

The Logger automatically records the following information about a Logger file:

- ♦ Protocol.
- ♦ Date and time of recording.
- ♦ Length of recording time.
- ♦ FOM file (HLA only).
- ♦ FOM modules (HLA Evolved only).

The Open File dialog box displays this information for any file you select, even if you cannot open the file using the version of the Logger that you are running. This is useful if your naming convention does not include the protocol used to record the file.

You can add a description of the recording to the file information.



Logger files created with Logger 3.x or earlier do not display basic information.

Logger files must use the Logger 5.3 or later format to store descriptive text. If you open an older file and do not upgrade, you will not be able to add descriptive text to the Logger information.

6.6.1. Displaying Information about a Logger File

You can display file information for the loaded Logger file at any time.

- To display file information, choose **File** → **Logger File Information**. The Logger File Information dialog box opens ([Figure 6-6](#)).

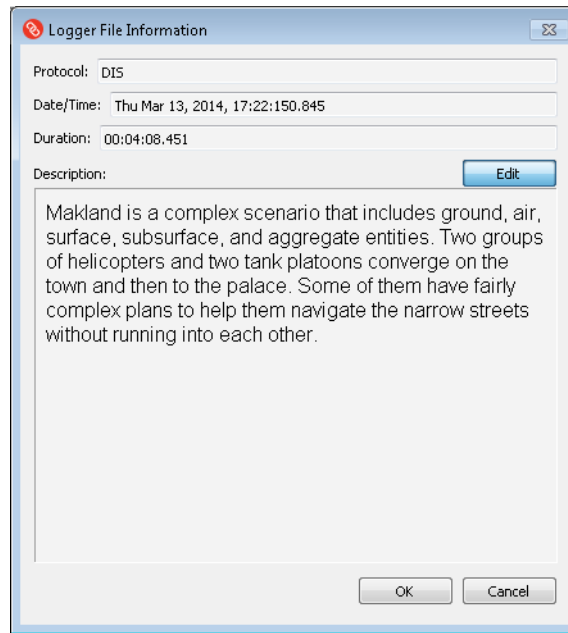


Figure 6-6. Logger File Information dialog box

6.6.2. Editing the Logger File Description

You can add text to a Logger file description to describe the contents or purpose of a recording, or any other useful information. The description uses HTML for formatting. If you type in text without HTML tags, it is displayed as an unformatted paragraph.

You can add a description while a file is open, when you first create it, or when you save it.

The basic file information (protocol, date/time, and so on) is read-only.

To edit the Logger file description while the file is open:

1. **File** → **Logger File Information**. The Logger Information dialog box opens ([Figure 6-6](#)).
2. Click **Edit**. The text area changes to edit mode.
3. Type the file description in the edit window. Use HTML tags to add formatting.
4. To preview the description, click **Preview**.
5. Click **OK**.
6. Save the Logger file.



The Logger preserves your changes to the description while the file is open. However, if you do not save the Logger file, your changes to the description are lost when you close it.

To add a file description when you start to record a file:

1. Choose **Record** → **Record As**. The New Logger File dialog box opens (Figure 6-7).

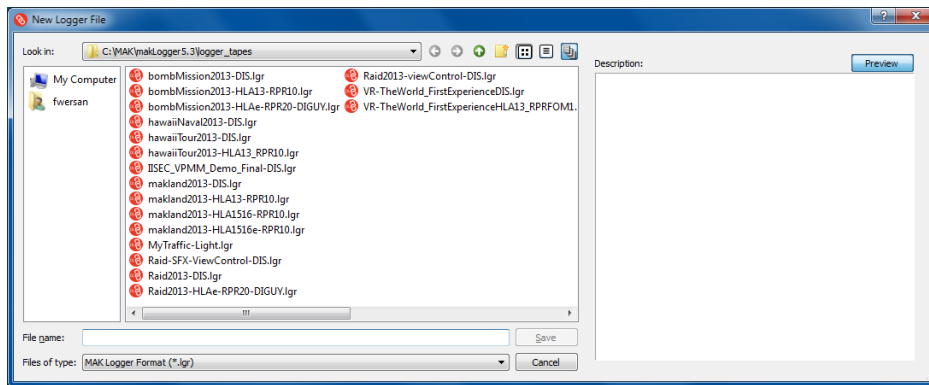


Figure 6-7. New Logger File dialog box

2. In the Description window, type the description. If you want it to be formatted, use HTML tags.
3. Optionally, click Preview to preview the formatted text.
4. Type a file name.
5. Click Save.

To add a file description when you save a file:

1. Choose **File** → **Save As**. The Save File As dialog box opens (Figure 6-8).

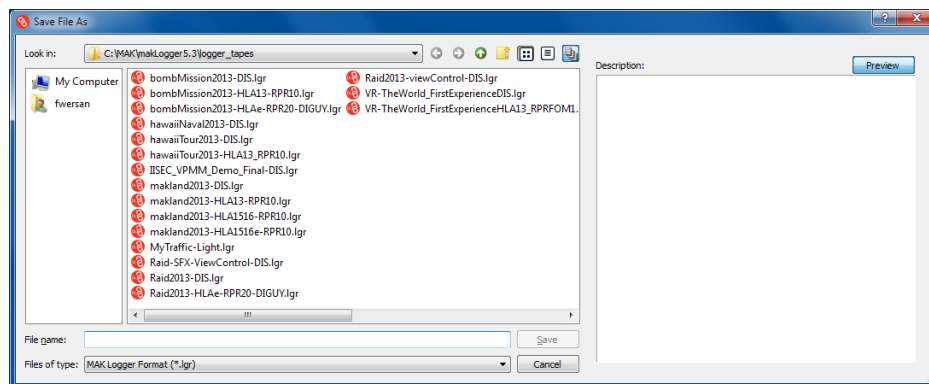


Figure 6-8. Save File As dialog box

2. In the Description window, type the description. If you want it to be formatted, use HTML tags.
3. Optionally, click Preview to preview the formatted text.
4. Type a file name.
5. Click Save.



7. Recording and Playing Logger Files

This chapter explains how to use the Logger's core features from the GUI.

Recording Exercises	7-3
Recording Immediately (to a Temporary File)	7-3
Recording to a Specified File	7-3
Recording to Multiple Files	7-4
Pausing the Recording Process	7-6
Stopping the Recording Process	7-6
Saving Files	7-7
File Backups	7-7
Saving a File to a New Name	7-7
Saving a Portion of a File to a New File	7-7
Playing Logger Files	7-8
Playing a Logger File	7-8
Pausing Playback of a Logger File	7-8
Playing a Segment of a File	7-8
Looping Playback	7-10
Stopping Playback of a Logger File	7-10
Setting the Playback Speed	7-10
Using High Speed Playback	7-11
Playing a File While it is Being Recorded	7-12
Zooming the Timeline	7-13
Zooming the Timeline Using the Mouse	7-13
Setting the Zoom from the Set Zoom Dialog Box	7-14
Moving the Zoomed Selection	7-14
Moving the Timepoint to a Specific Time	7-15

Jumping to the Beginning or End of the Playback Range	7-15
Moving to a Specific Time in a File.....	7-15
Setting the Time Using the Timeline Cursor	7-15
Viewing Packet Histograms	7-16
Scaling the Packet Count View	7-17
Viewing Object Statistics	7-18

7.1. Recording Exercises

When you record an exercise, you can specify a name for the new recording and then start the recording process, or you can start recording immediately and save the recording to a file when you are done (the default behavior).

The Status Bar displays the starting time, the current time, and the total time of the recording.

As recording progresses, the Time field is updated to display elapsed time. By default, the zoom timeline shows the most recent 30 seconds of the recording. (You can change this with the `SetFollowingInterval` command.) The full timeline shows the entire length of the recording. In record mode, the timescale is fixed at 1.0 (normal speed).



- ♦ If you are recording an exercise that is running faster than real-time, the Logger records the simulation at that rate and will play it back faster than real-time when the playback timescale is set to 1.0.
 - ♦ You cannot record and play back a file at the same time using a single Logger.
-

7.1.1. Recording Immediately (to a Temporary File)

- To begin recording, choose **Record** → **Record**, or click the Record button (●).

7.1.2. Recording to a Specified File

Unless you enable Pause on Record As, the Logger starts recording as soon as you specify a filename.

To record an exercise to a named file:

1. Choose **Record** → **Record As**. The New Logger File dialog box opens.
2. Type a new filename or select an existing file.



If you select an existing file, its contents will be overwritten by the new file.

3. Click Save. If Pause on Record As is disabled (the default), the Logger starts recording immediately.
- If Pause on Record As is enabled, to begin recording, choose **Record** → **Record**, or click the Record button (●).

Enabling Pause on Record As

By default, when you choose Record As, the Logger starts recording as soon as you close the New Logger File dialog box. If you do not want to start recording immediately, you must enable Pause on Record As. When Pause on Record As is enabled, the Logger does not start recording until you click the Record button.

- To enable or disable immediate recording for the Record As command, choose **Options** → **Pause on Record As**.

7.1.3. Recording to Multiple Files

The Logger can record an exercise to multiple files as it is recording the primary Logger file. When you record to multiple files, the Logger saves data to file at specified time intervals. You can play or otherwise process these files in a second Logger while the first Logger continues to record the exercise. This option also ensures that if logging of the exercise is interrupted due to system or application instability, you will not lose the data saved prior to the application problems.

To record to multiple files:

1. Choose **Record** → **Record to Multiple Files**. The Record to Multiple Files dialog box opens.
2. Optionally, change the location to which the complete file will be saved in the Complete File Directory text box.
3. In the Complete File Name box, specify the name of the complete file.

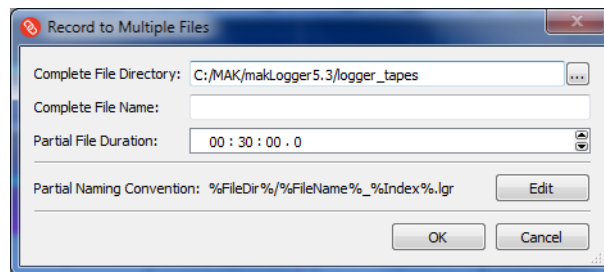


Figure 7-1. Record to Multiple Files dialog box

4. Specify the length of each partial file in the Partial File Duration box.
5. Optionally, change the convention used to name the new files. For details about changing the partial file naming convention, please see [“Specifying the Partial File Naming Convention,”](#) on page 7-5.
6. Click OK.

Specifying the Partial File Naming Convention

The default convention for naming partial files is *completeDirectoryPath_filename_n.lgr*. Files are written to the complete directory path. However, you can change the naming convention using a set of macros provided with the Logger.

To change the partial file naming convention:

1. Choose **Record** → **Record to Multiple Files**. The Record to Multiple Files dialog box opens.
2. Click Edit. The dialog box expands to display a list of macros (Figure 7-2).

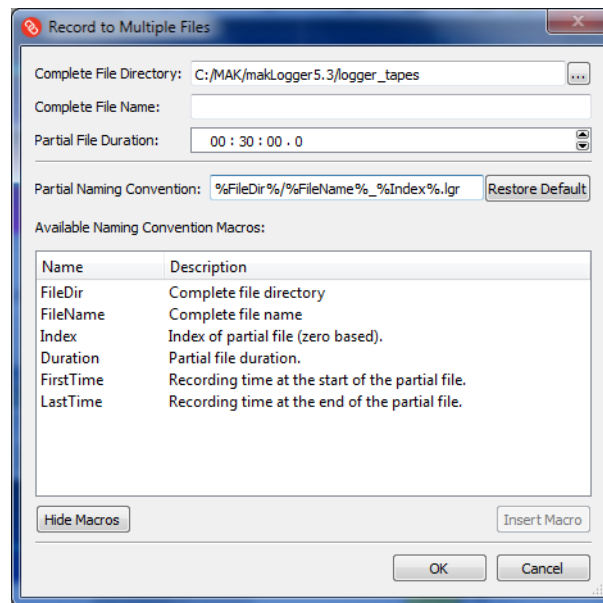


Figure 7-2. Record to Multiple Files dialog box in Edit mode

3. Edit the text in the Partial Naming Convention box manually as desired or add macros, or both.
4. To add a macro to the naming convention, select it in the list and click Insert Macro (or double-click the list entry). It gets added at the location of the insertion point in the Partial Naming Convention box. To ensure that each partial file has a unique name, the naming convention must include at least one of Index, First-Time, or LastTime. For explanations of the macros, please see Table 7-1.
5. Click Hide Macros to close the macro list.
6. Click Restore Default to revert to the default naming convention.

Table 7-1: Partial file naming convention macros

Macro	Definition
FileDir	The path of the directory in which the complete file will be saved. The macro does not include the final path separator.
FileName	The name of the complete file, without the file extension.
Index	An integer index, starting at 0 and increasing by one for each partial file created.
Duration	The duration of the file in seconds. Each partial file, except the final file will be this long.
FirstTime	The time in the complete file that maps to the beginning of the new file, in seconds.
LastTime	The time in the complete file that maps to the end of the new file, in seconds.

7.1.4. Pausing the Recording Process

You can pause the recording process and then resume recording to the same file.



When you pause during the recording process, the file is not saved. You cannot play back the file from the paused recording state. If for some reason you exit the Logger while in the paused state, the file will not be saved.

- To pause the recording process, click the Pause button (⏸).
- To resume a paused recording, choose **Record** → **Record**, or click the Record button (⏺).

7.1.5. Stopping the Recording Process

When you stop recording, the Logger enters playback state and you cannot resume recording. If you want to resume a recording, pause it.



If the Logger has not recorded any data, it cannot create a file.

- To stop recording, choose **Record** → **Stop**, or click the Stop button (⏹).

7.2. Saving Files

When you record a simulation or edit a Logger file, the file is in an unsaved state, which is indicated by an asterisk next to the filename in the Title bar. If you specify a name for a file before you start the recording process, it is automatically saved when you stop recording. If you have an unsaved file loaded, the Logger prompts you to save it before exiting, opening a file, closing a file, or recording a new file.

- To save a file, choose **File** → **Save**. If the file already has a filename, it is saved immediately. If it does not yet have a filename, you are prompted to supply one (Save As).

7.2.1. File Backups

When you save a previously saved file, the Logger creates a backup file by renaming the previous version of the file to *filename.bak.lgr*. For example, if you edit *maklanddemoDIS.lgr* and save the changes, the original version of *maklanddemoDIS.lgr* is saved to *maklanddemoDIS.bak.lgr*.

The Logger maintains only one backup for a particular Logger file. If you edit and save a file that was previously backed up, the original backup file gets replaced with a newer backup file. For example, if after saving *maklanddemoDIS.lgr* in the previous example, you were to edit it again and save it, a new version of *maklanddemoDIS.bak.lgr* would replace the original backup file.

You can disable automatic backup with the `SetBackupOnSave` command.

7.2.2. Saving a File to a New Name

- To save a file to a new name, choose **File** → **Save As** and type a new name in the Save As dialog box.

7.2.3. Saving a Portion of a File to a New File

You can save a segment of a file to a new file. The timestamps of the packets in the new file are offset so the first packet is at time zero. If the segment has annotations, they get saved with the segment.

To save a segment of a file to a new file:

1. Open the file.
2. Use the timeline sliders to select the segment you want to save. (The selected segment is the white area of the time line, not the gray area.)
3. Choose **File** → **Export Selection To** → **Logger File**. A standard Save file dialog box opens.
4. Enter a name for the new file.
5. Click OK.

7.3. Playing Logger Files

The Status Bar in the Logger window displays the starting time for playback, the current time in the file, and the ending time for playback.



- ♦ Playback options are restricted in time-managed mode. For details, please see [“Some Logger Features are Restricted When You Use Time Management,”](#) on page 3-5.
- ♦ For information about how to configure the Logger for faster playback, please see [“Configuring Faster Playback,”](#) on page 5-5.

7.3.1. Playing a Logger File

By default, the Logger enters a stopped state when a file is opened.

To play a Logger file:

1. Open the file.
2. Choose **Play** → **Play/Pause**, or click the Play button (

Enabling Immediate Playback

When Play On Load is enabled, the Logger starts playing as soon as you open a file. If the file has an annotation file, it is loaded automatically. When Play On Load is disabled, you must click the Play button to start playing.

- To enable or disable immediate playback, choose **Options** → **Play On Load**.

7.3.2. Pausing Playback of a Logger File

When you pause playback, the Logger keeps the current velocity and acceleration values for each simulation object, but sets the Frozen bit so simulation objects do not move. By default, the dead-reckoning algorithm is set to static. For details about setting the dead reckoning algorithm, please see [“Setting the Dead Reckoning Algorithm on Pause,”](#) on page 5-9.

- To pause playback of a Logger file, choose **Play** → **Play/Pause**, or click the Pause button (

7.3.3. Playing a Segment of a File

You can configure the Logger to play any arbitrary segment of a file. If you have the Logger configured to loop, it loops within the segment you have specified. You can specify exact start and stop times in a dialog box or you can specify the start and end times with the time sliders.

Specifying a Playback Segment Using the Time Sliders

To play a segment of a file:

1. On the Logger timeline, drag the start time slider (Figure 7-3) to the time at which you want to start playback. The Start time display in the Status Bar shows the exact time that you select.

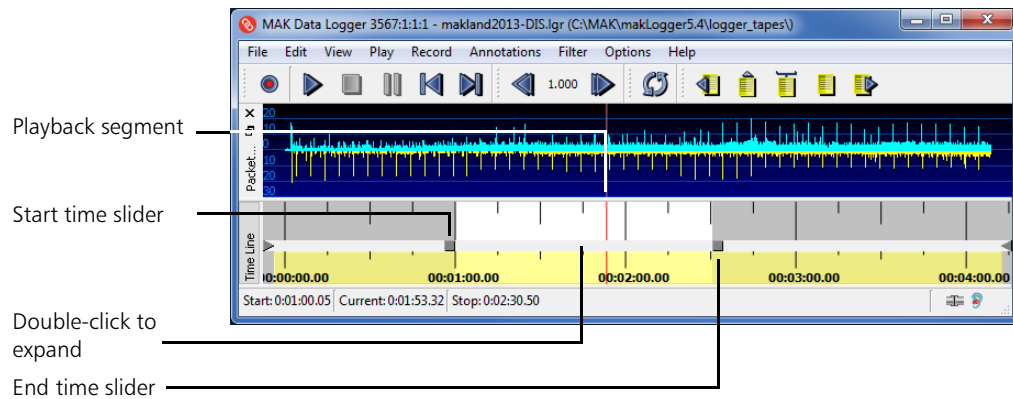


Figure 7-3. Timeline sliders

2. Drag the end time slider to the point on the timeline at which you want to end playback. The Stop time display in the Status Bar shows the exact time that you select.
3. Play the file.

Specifying a Playback Segment in a Dialog Box

To specify a playback segment using a dialog box:

1. Choose **Play** → **Set Start and End Time**. The Set Start and End Time dialog box opens (Figure 7-4).

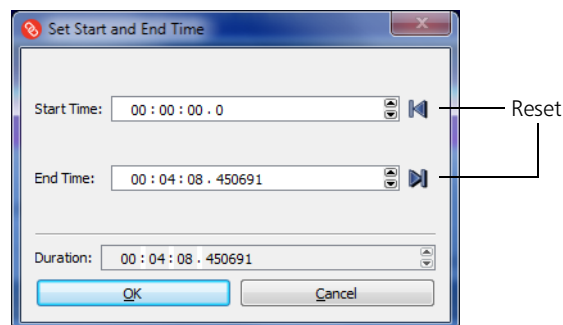


Figure 7-4. Set Start and End Time dialog box

2. Type start and end times or use the spinner control to select them. The duration of the segment is displayed in the Duration box.
3. To reset the start time to 0 or the end time to the end of the file, click the reset buttons to the right of the time value.
4. Click OK.

Expanding the Playback Segment to Full Length

- To expand the playback segment to the full length of the file, drag the sliders back to the edge of the window or double-click the slider bar between the start and end time sliders (Figure 7-3).

7.3.4. Looping Playback

When the Logger plays a file to the end or the end of a playback range, it can stop or it can loop back to the beginning of the file or range and start playing it again.

- To set the looping behavior, choose **Play** → **Toggle Loop at End**, or click the Toggle Loop At End icon (🔄) in the Logger window (Figure 6-1).

7.3.5. Stopping Playback of a Logger File

- To stop playback of a Logger file, choose **Play** → **Stop**, or click the Stop button (■).

7.3.6. Setting the Playback Speed

You can vary the playback speed. The Speed toolbar lets you increase speed by a factor of 2 and decrease by 0.5. The Set Playback Speed dialog box lets you increase and decrease the speed by any value. If you set the speed using the Set Playback Speed dialog box, the Speed toolbar displays the new speed and increases or decreases the speed relative to the new value, not the default set of values. The actual speed of the action in the file depends on the speed at which you recorded it.

To set the playback speed:

1. Choose **Play** → **Playback Speed** → **Set Playback Speed**, or click the current speed setting on the Speed toolbar. The Set Playback Speed dialog box opens (Figure 7-5).

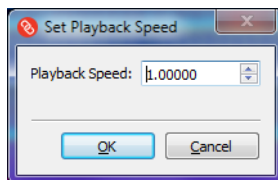


Figure 7-5. Set Playback Speed dialog box

2. Type a speed or set it using the controls.
3. Click OK.

You can quickly increase or decrease the playback speed by a factor of two.

- To quickly increase or decrease playback speed, choose **Play** → **Playback Speed** → **Increase Playback Speed 2x** (or **Decrease Playback Speed 1/2x**), or click the Increase Speed button () or Decrease Speed button () on the Speed toolbar.

7.3.7. Using High Speed Playback

The high speed playback feature lets you play files at higher speeds than would normally be possible. (The Logger has a practical limit on the rate at which it can send updates and interactions based on processor and network speed.) Use it when the combination of playback speed and file density (number of recorded packets per second) results in a required packet output rate that exceeds the rate at which the Logger can transmit the packets on the network. For example, high speed playback would be useful for simulations with many entities, or simulations that are quite lengthy (for which you might want to speed up playback significantly), or both.

In high speed playback, instead of playing every packet in the file, the Logger skips from checkpoint to checkpoint and sends the entity updates for the entities that exist at the time of each checkpoint. Interactions get sent if they were saved to the checkpoints. Where necessary, the Logger creates or removes the entities that were created or removed in between the jumps. The effect is very similar to that of a digital video player 20-60x playback, in which snapshots of scenes are shown, but not every frame.

When high speed playback is enabled, the Logger sends updates only for checkpoints. It does not send any messages between checkpoints. The frequency of updates is determined by the frequency of checkpoints in the file and the playback speed. For example, suppose that you have a file that has checkpoints every 60 seconds. If you play the file at 60x speed, playback would advance to a new checkpoint every second.



To use high speed playback, a file must contain checkpoint data.

By default, high speed playback is disabled. To enable it, you must edit the *lgrConfig.xml* file. In *lgrConfig.xml*, set the value for the `skippingPlaybackThreshold` variable to a value greater than 0, for example:

```
<var name="skippingPlaybackThreshold" type="double" value="16.0"/>
```

The value you enter is the playback speed at or above which the Logger skips from checkpoint to checkpoint rather than playing every packet. Therefore, do not set the threshold too low unless you have a very high entity count.

Once you configure high speed playback, set the playback speed, as described in [“Setting the Playback Speed,”](#) on page 7-10, and if the speed meets or exceeds the threshold, high speed playback is used.

7.3.8. Playing a File While it is Being Recorded

You can play a file as it is being recorded. You need two instances of the Logger to do this. One records the exercise. The second plays the file that is being recorded.

When you open a file that is being recorded, the file is incomplete. So the Logger prompts you to repair it. If you repair the file, it is saved and opened as a complete file. (For more information about incomplete files and file repair, please see “[Opening Incomplete Files](#),” on page 6-9.) If you do not repair the file, the Logger opens the file for playback. The file duration matches the file length at the time it was opened. When you start playing the file, the Logger continuously updates the end of the file to capture the additional data as it is being recorded so that it is available for playback.

Since the file you are playing back is incomplete, some data is not available, such as the index table for efficient jumping within the file.

To play a file as it is being recorded:

1. Start two Loggers.
2. Begin recording the exercise using one of the Loggers.



It is recommended that you record the file using **Record** → **Record As**, so that the file has a meaningful name and is in the `./logger_tapes` directory (or wherever you decide to save it). If you record without using Record As, the file is saved to a temporary file. In Windows 7, this is `C:\Users\<user>\Local\Temp\lgr*.tmp`.

3. In the second Logger, choose **File** → **Open**. The Open File dialog box opens.
4. Select the file that you are recording. If you did not name the file when you started to record it, you must find the file in a temp directory. You are prompted to repair the file.
5. To play only the portion of the file that has been recorded to the point at which you opened it, click Yes.

To play the file continuously as it is recorded, click No.

6. Play the file.

7.4. Zooming the Timeline

You can zoom in and out on the timeline (adjust the granularity). You can zoom using the mouse or you can specify zoom boundaries, center point, and duration in the Set Zoom dialog box.

Figure 7-6 shows the zoom timeline zoomed to the range from approximately 1:30 to 2:45. The full timeline shows that range in yellow. The timeline cursor is at the same time point in each timeline. The Packet Count View is zoomed to show only the packets during the same time range as the zoom timeline.

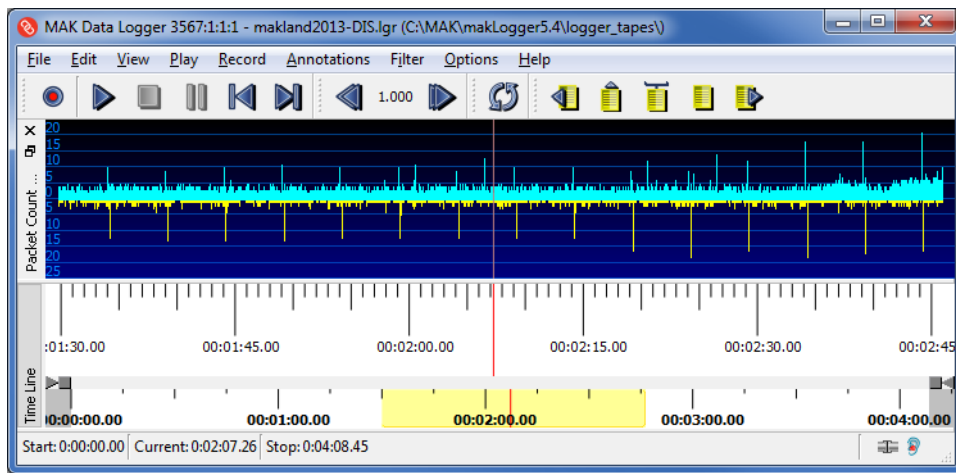


Figure 7-6. Zoomed timeline

7.4.1. Zooming the Timeline Using the Mouse

To zoom the timeline with the mouse:

1. Move the mouse cursor over the timeline.
2. Move the mouse wheel forward to zoom in. Move it backwards to zoom out.

7.4.2. Setting the Zoom from the Set Zoom Dialog Box

The Set Zoom dialog box allows you to precisely specify the start and end point of the zoomed view.

To zoom the timeline using the Set Zoom dialog box:

1. Choose **View** → **Set Zoom**. The Set Zoom dialog box opens ([Figure 7-7](#)).

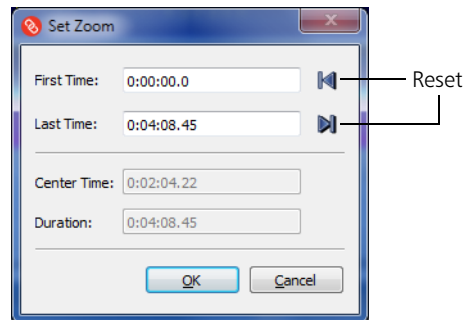


Figure 7-7. Set Zoom dialog box

2. Enter the beginning time in the First Time box.
3. Enter the end time in the Last Time box.
4. If you want to reset the start time value to 0:00:00.0 or the end time to the end of the file, click the appropriate Reset button.
5. Click OK.

7.4.3. Moving the Zoomed Selection

If you have zoomed the timeline and want to look at a different portion of it at the same zoom level, you can easily move the time range.

- To change the time range for the zoomed selection, put the mouse cursor over the yellow segment of the full timeline and drag it to the time range you want to look at.

7.5. Moving the Timepoint to a Specific Time

The Logger provides the following ways to move the timepoint to a specific time:

- ♦ Move to the beginning or end of the playback range.
- ♦ Specify a timepoint in a dialog box or using the mouse.
- ♦ Move to a time defined by an annotation.

i

- ♦ If you execute a time jump while playback is paused or stopped, the Logger remains in that state. The Logger also retains a play state after a time jump unless the end of the file is reached and wrapping is disabled.
 - ♦ During slow-speed playback, when you use the Set Time option, the Logger sends interim messages to ensure that entities do not time out. For more information, please see [“Updating Entity Information Quickly After a Time Jump,”](#) on page 3-7.
-

7.5.1. Jumping to the Beginning or End of the Playback Range

- To jump to the beginning of a playback range, choose **Play** → **Jump To Start**, or click the Jump to Start button (.
- To jump to the end of a playback range, choose **Play** → **Jump To End**, or click the Jump to End button (.

7.5.2. Moving to a Specific Time in a File

The Set Time feature lets you move to a specific point in time measured from the start of the file. The speed of time jumps is affected by the checkpoint policy. For details about setting the checkpoint policy, please see [“Configuring Indexing,”](#) on page 5-3.

To move to a specific time:

1. Choose **Play** → **Jump to Time**. The Jump to Time dialog box opens.
2. Enter the time you want to jump to.
3. Click OK.

7.5.3. Setting the Time Using the Timeline Cursor

The current timepoint during playback is indicated by a red bar (cursor) that moves along the timeline ([Figure 6-1](#)).

To set the playback timepoint using the timeline cursor, do one of the following:

- ♦ To set a timepoint within the zoom range, click a location in the zoom timeline.
- ♦ To set a timepoint outside of the zoom range, click on the full timeline.

7.6. Viewing Packet Histograms

You can display a histogram of the number of packets being sent or recorded by the Logger (Packet Count View) and their size (Packet Size View). When you load a file with the Packet Count or Packet Size views enabled, the Logger calculates the histogram values for the entire file. Then it displays them (Figure 6-1). When you record a simulation with these views enabled, the Logger updates them as data is recorded (Figure 7-8).

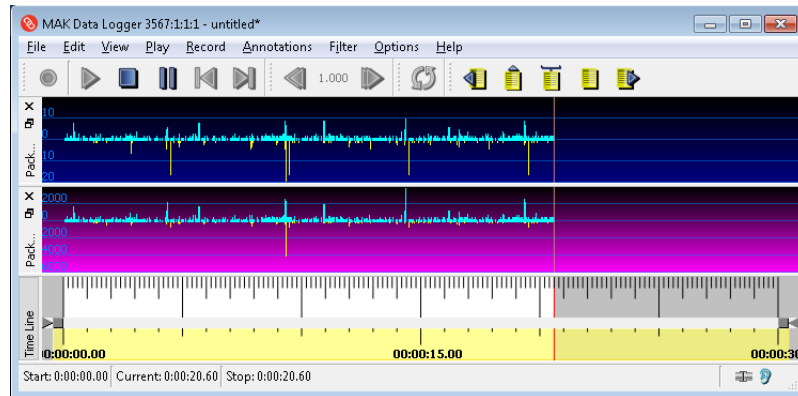


Figure 7-8. Packet histograms in record mode

In the packet views, blue lines (above the centerline) represent the object packets; yellow lines (below the centerline) represent the interaction packets; red lines (small tick marks along the centerline) represent checkpoints.

Each line represents the count or size over a small interval of time. You can adjust the length of the interval with the `SetPacketViewInterval` command. For more information, please see Chapter 13, *Using Logger Commands*.

i

- ◆ Since the Logger updates the packet views when it loads a file, loading large files with these views visible will increase load time considerably.
- ◆ The packet views are completely updated each time you edit a file.

- To show or hide the Packet Count View histogram, choose **View** → **Packet Count View**.
- To show or hide the Packet Size View histogram, choose **View** → **Packet Size View**.

7.6.1. Scaling the Packet Count View

You can change the scale of the Packet Count View (Figure 7-9). When you zoom in, a scroll bar is added to the right end of the view. You can move the histogram up or down in the window to see the parts that have scaled out of the view.

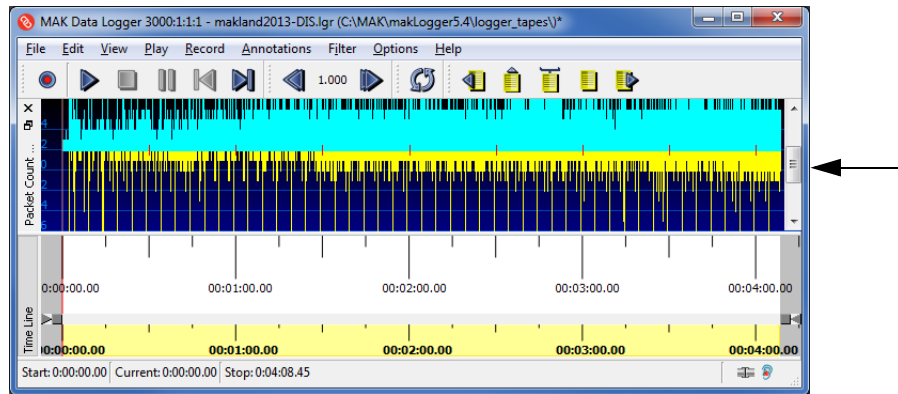


Figure 7-9. Scaled Packet Count View

To change the scale of the Packet Count View:

1. Move the mouse over the Packet Count View.
2. Move the mouse wheel to zoom in and out.
3. Optionally, move the scroll box up or down to move the packet count up or down in the window.

7.7. Viewing Object Statistics

The Logger can display statistics about the objects being published (during playback) or reflected (during recording) by the Logger application (Figure 7-10).

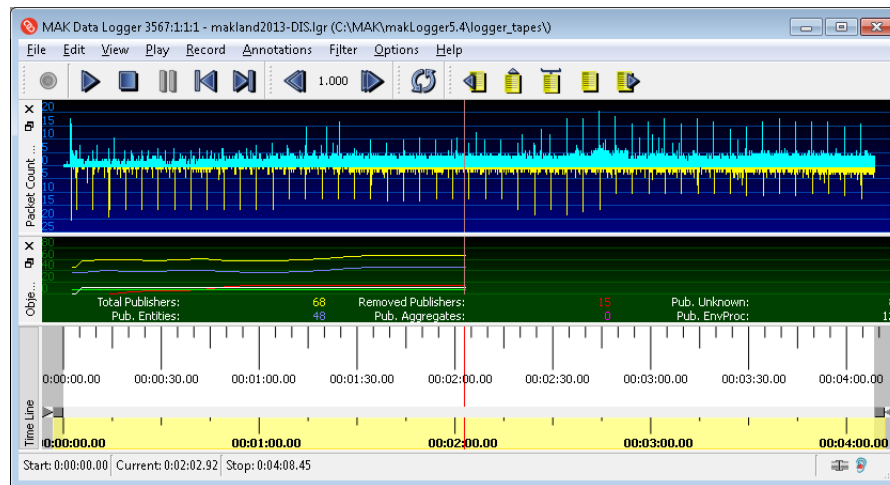


Figure 7-10. Object statistics

- To view or hide object statistics, choose **View → Object Statistics**.



8. Annotating Files

This chapter explains how to add annotations to Logger files.

Annotating a Logger File.....	8-2
Adding an Annotation.....	8-2
Saving an Annotation File	8-4
Opening an Annotation File.....	8-5
Importing an Annotation File	8-5
Selecting an Annotation	8-5
Editing an Annotation.....	8-5
Deleting an Annotation.....	8-6
Moving the Timeline Cursor to the Start of an Annotation.....	8-6
Jumping to the Next or Previous Annotation	8-7
Zooming the Timeline to an Annotation.....	8-8
Bounding Playback to an Annotation	8-9

8.1. Annotating a Logger File

An annotation is associated with a particular time or time range in a Logger file. Each annotation has a name, and you can associate text with it. Besides being a way to store information about the events in a Logger file, annotations allow you to quickly zoom in on a file or bound playback to a segment of a file. The Annotations View has a context sensitive menu from which you can execute the various annotation commands.

8.1.1. Adding an Annotation

To annotate a Logger file:

1. Choose **View** → **Annotations**. Annotations, if any, are displayed in the Logger data display area (Figure 8-1).

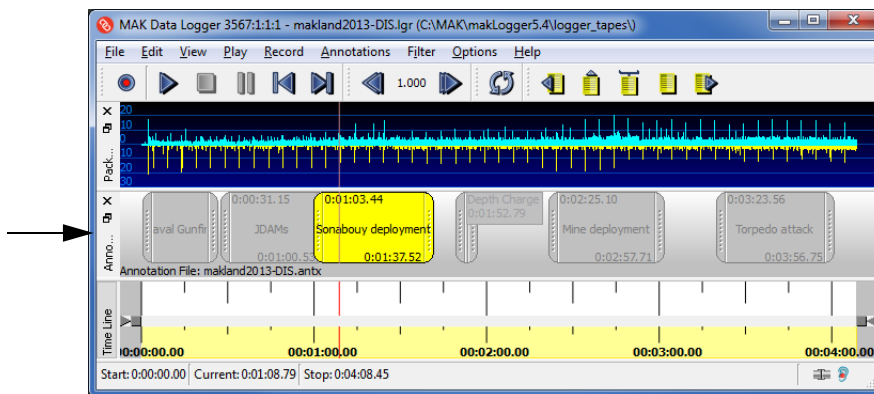


Figure 8-1. Annotations display

2. Choose **Annotations** → **Create** → *annotation_type*, where *annotation_type* type is Point Annotation, Range Annotation, or an instant annotation. (For details about inserting instant annotations, please see “Adding an Instant Point Annotation,” on page 8-3.) The Point Annotation (or Range Annotation) dialog box opens.

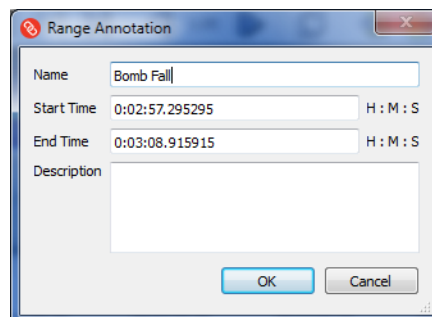


Figure 8-2. Range Annotation dialog box

3. In the Name text box, type a name for the annotation. The name is displayed in the annotation view.
4. Optionally, change the timepoint. For a point annotation, the default time for the annotation is the current timepoint. For a range annotation, the default start time and end time for the annotation's time range is the current timepoint. The time is specified in hour:minute:second.tenth_of_second format.
5. Optionally, in the Description box, enter any text that you want to accompany the annotation.
6. Click OK. The annotation is added to the Annotations window ([Figure 8-1](#)).



If you create an annotation while the Annotations View is hidden, the Annotations View is automatically displayed.

Adding an Instant Point Annotation

Instant annotations add an annotation at the current time point without requiring you to fill out a dialog box. The annotation time, for point annotations, or start time, for range annotations, is set to the time point. This allows you to quickly add annotations as you play or record a file without needing to stop the file. You would typically go back and edit the annotation at a later time.

Instant range annotations are inserted with identical start and end times.

To add an instant annotation, do one of the following:

- ♦ Choose **Annotations** → **Create** → **Instant Point Annotation** (or **Instant Range Annotation**).
 - ♦ Press \ (backslash) for a point annotation or [(open square bracket) for a range annotation.
 - ♦ Click the Add Point Annotation button () or the Add Range Annotation button () .
- To set the end time for an instant range annotation, use the Update Annotation feature described in the next section or press the] key at the point you want to end the annotation's range.

Dynamically Resetting an Annotation's Start or End Time (Update)

You can easily move a point annotation or change the start or end time of range annotation. The Update Annotation command lets you change the start and end times of the currently selected annotation. (This procedure works while the Logger is playing, recording, or paused.)

When you update a point annotation, it moves to the new time. When you update a range annotation, if the current time is earlier than the start time, the start time is reset. If the current time is later than the start time, the end time is reset.

To update an annotation:

1. Create an annotation or select an annotation.
2. If the recording is paused, click on the zoom timeline at the time point to which you want to update the annotation. (If you are recording or playing a recording, Update uses the current time in the recording.)
3. Choose **Annotations** → **Update**, press] (right bracket), or click the Update Annotation button (.

8.1.2. Saving an Annotation File

You can save annotations to a file. Annotation files have the extension *.antx*.

To save annotations to a file:

1. Create annotations as described in [“Adding an Annotation,”](#) on page 8-2.
2. Choose **Annotations** → **Save**. The Save Annotation File dialog box opens.
3. Type a name for the annotation file.
4. Click Save.

8.1.3. Opening an Annotation File

When you open an annotation file, any annotations currently displayed in the Logger window are removed and replaced with the annotations in the file you opened. If you load a file with Play on Load enabled and the file has an annotation file, the annotation file is loading automatically.



You can open annotation files that have been saved in legacy format (*.ant*) and the current format (*.antx*).

To open an annotation file:

1. Choose **Annotations** → **Open**. The Open Annotation File dialog box opens.
2. Select an annotation file.
3. Click Open. The annotations in the file are displayed.

8.1.4. Importing an Annotation File

When you import an annotations file, the annotations are added to any annotations that are already displayed in the Logger window.

To import an annotation file:

1. Choose **Annotations** → **Import**. The Open Annotation File dialog box opens.
2. Select an annotation file.
3. Click Open. The annotations in the file are imported.

8.1.5. Selecting an Annotation

- To select an annotation, choose **Annotations** → **Select** → *annotation*, where *annotation* is one of the annotations in the list, or in the Annotations View, click the annotation.

8.1.6. Editing an Annotation

You can edit the properties of an annotation (name, text, and start and end times).

To edit an annotation:

1. Select the annotation and choose **Annotations** → **Properties**, or double-click the annotation. The Point Annotation or Range Annotation dialog box opens.
2. Edit the properties as desired.
3. Click OK.

Editing a Point Annotation Directly

- To edit the time of a point annotation, drag it to a new time.

Editing a Range Annotation Directly

You can edit the start and end times of a range annotation by dragging its endpoints or dragging the entire annotation.

To edit a range annotation use either or both of the following methods:

- ♦ In the Annotations bar, click and hold the left mouse button on an end point of an annotation and drag it to a new time.
- ♦ In the Annotations bar, click and hold the left mouse button anywhere inside an annotation's range and drag the entire annotation to a new location.

Converting an Annotation to the Other Annotation Type

You can convert point annotations to range annotations and range annotations to point annotations. The time point of a point annotation becomes the start time of the new range annotation. You can then change the end time as described in previous sections. The start point of a range annotation becomes the time point of a point annotation.

To convert an annotation to the other annotation type:

1. Select the point annotation that you want to convert.
2. Choose **Annotations** → **Convert To** → **Range Annotation** (or **Point Annotation**).
3. If you converted a point annotation to a range annotation, specify the end point of the annotation, as described in [“Editing an Annotation,”](#) on page 8-5.

8.1.7. Deleting an Annotation

To delete an annotation:

1. Select the annotation you want to delete.
2. Choose **Annotations** → **Delete**, or press the **Delete** key.

If you remove an annotation from a set of annotations that you have saved to a file, save the edited list with the **File** → **Save** command.

8.1.8. Moving the Timeline Cursor to the Start of an Annotation

You can move the timeline cursor to an annotation or the start time of a range annotation.

To move the timeline cursor to an annotation:

1. Select the annotation.
2. Choose **Annotations** → **Jump to Start**.

Moving the Timeline Cursor to the End of an Annotation

You can move the timeline cursor to the end time of a range annotation.

To move the cursor to the end of an annotation:

1. Select the annotation.
2. Choose **Annotations** → **Jump to End**.

8.1.9. Jumping to the Next or Previous Annotation

You can quickly jump to the start of the next or previous annotation, if there is one.

- To jump to the previous annotation, choose **Annotations** → **Jump to Next**, or click the Jump to Next Annotation button (.
- To jump to the previous annotation, choose **Annotations** → **Jump to Previous**, or click the Jump to Previous Annotation button (.

8.1.10. Zooming the Timeline to an Annotation

You can zoom the timeline view to an annotation. For example, in [Figure 8-3](#), in the first view, the zoom timeline shows the entire length of the file. After zooming to the Sonobuoy deployment annotation, the zoom timeline is zoomed to the start and end points of the annotation.



Zooming the timeline to an annotation does not change the location of the timeline cursor.

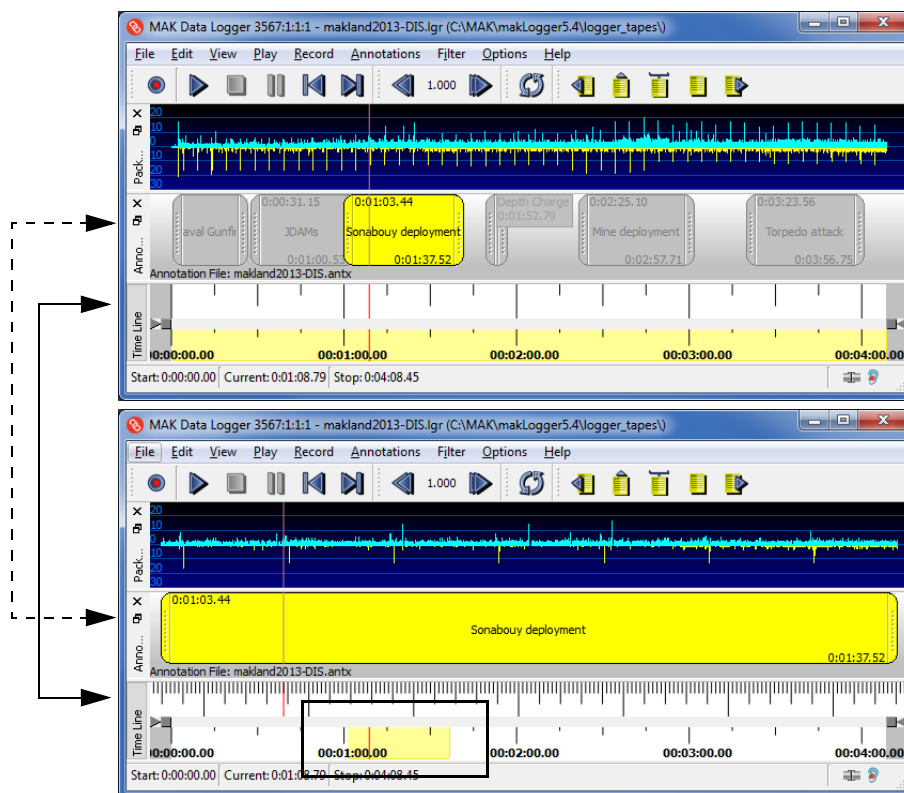


Figure 8-3. Zooming to an annotation

To zoom the timeline to an annotation:

1. Select the annotation.
2. Choose **Annotations** → **Zoom to Annotation**. The Annotations bar and the time line zoom to fit the range of the annotation ([Figure 8-3](#)).

8.1.11. Bounding Playback to an Annotation

You can set the playback segment to the range of an annotation. For example, in [Figure 8-4](#), in the first view, the playback handles are at the start and end of the file. After bounding to the annotation, the playback handles are at the start and end points of the annotation. For further comparison, notice that in [Figure 8-3](#), the playback handles are not affected by zooming to the annotation.

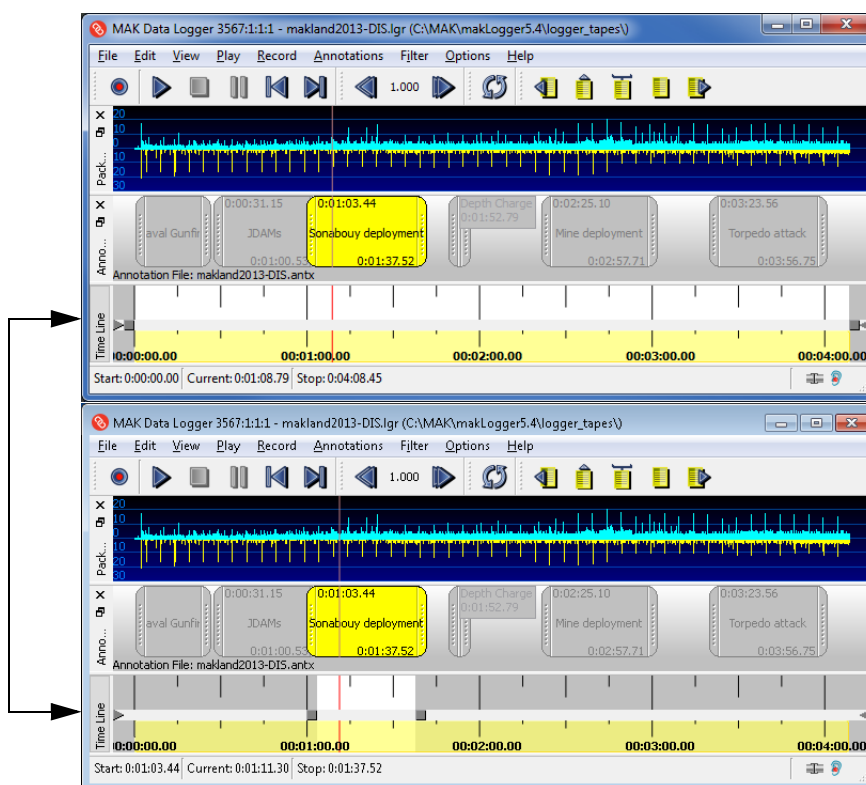


Figure 8-4. Bounding playback to an annotation

To set the playback segment to the range of an annotation:

1. Select the annotation.
2. Choose **Annotations** → **Bound to Annotation**. The playback range changes to match the start and end points of the annotation ([Figure 8-4](#)).



9. Filtering Input and Output

This chapter explains how to filter input to the Logger during recording and output during playback.

Filtering Logger Input and Output	9-2
Filtering by Entity Type.....	9-2
Removing Entity Types from the Entity Types Lists	9-3
Configuring the List of Common Entity Types	9-3
Filtering DIS PDUs by PDU Kind	9-4
Including Custom PDUs in Filter Lists.....	9-5
Filtering DIS Messages by Entity ID.....	9-6
Filtering HLA Messages by Object Name	9-7
Filtering HLA Exercises by Class.....	9-9
Filtering VR-Forces Messages	9-10

9.1. Filtering Logger Input and Output

When you record an exercise or play a file, you can filter the input or output. In DIS, you can filter by entity type, entity ID, or PDU. In HLA, you can filter by class or object name. During a Logger session, filtering settings apply to both file and playback. Settings do not persist between sessions.



The Logger does not apply filters when you export to a database.

9.2. Filtering by Entity Type

You can filter DIS PDUs or HLA update messages by entity type. The procedure is the same in all versions of the Logger.

To filter Logger input or output by entity type:

1. Choose **Filter** → **By DIS Entity Type** (or for HLA, **By Entity Type**). The Filter by Entity Type dialog box opens (Figure 9-1). If common entity types are configured in the Logger configuration file or were specified on the command line when the Logger was started, they are listed in the window. If any of them were configured to have filtering enabled, the check box in the Filtered column is selected.

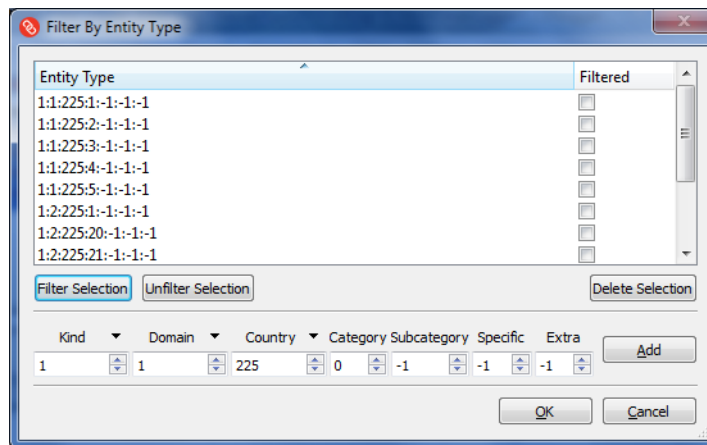


Figure 9-1. Filter By Entity Type dialog box

If the entity type that you want to filter is not listed in the list:

- a. Enter the 7-digit enumeration for the entity type you want to filter in the boxes at the bottom of the dialog box.
 - b. Click Add. The entity type is added to the list.
2. To filter out an entity type, check the box in the Filtered column for that entity.
 3. To process a filtered entity, clear its check box in the Filtered column.

4. To filter or unfilter multiple entities at the same time, select the entities in the list and click Filter Selection or Unfilter Selection.

9.2.1. Removing Entity Types from the Entity Types Lists

- To remove an entity type from the Entity Types lists, select the entity type and click the Delete Selection button.

9.2.2. Configuring the List of Common Entity Types

You can configure a list of common entity types to populate the Common Entity Types list in the Filter by Entity Type dialog box. To configure a list of entity types, use the `SetCommonEntityTypes` command on the command-line or text interface, or add entries to the `commonEntityTypes` variable in *lgrConfig.xml*, as in the following example:

```
<var name="commonEntityTypes" type="list" listtype="string">
  <!-- US tank -->
  <listitem>1:1:225:1:-1:-1:-1</listitem>
  <!-- US armored vehicle -->
  <listitem>1:1:225:2:-1:-1:-1</listitem>
  <!-- US armored utility vehicle -->
  <listitem>1:1:225:3:-1:-1:-1</listitem>
  <!-- US self-propelled artillery -->
  <listitem>1:1:225:4:-1:-1:-1</listitem>
</var>
```

Listing an entity type using `commonEntityTypes` or `SetCommonEntityTypes` does not enable filtering of that entity type. If you just use one of these methods for listing common entity types, then to filter them, you must enable filtering in the GUI, as described in “Filtering by Entity Type,” on page 9-2. If you want to automatically enable filtering for a common entity type, you can do so with the `filterEntityTypes` variable in *lgrConfig.xml* or the `FilterEntityTypes` command. The syntax is the same as for the common entity types variable and command.

If you specify an entity type using `filterEntityTypes` or `SetFilterEntityTypes`, when you open the Filter By Entity Type dialog box, the entity type is listed and the check box in the Filtered column is selected.

The two lists can overlap. Whichever command executes last or comes last in the configuration file takes precedence.

9.3. Filtering DIS PDUs by PDU Kind

To filter Logger input and output by PDU kind:

1. Choose **Filter** → **By DIS PDU Kind**. The Filter by PDU Kind dialog box opens (Figure 9-2).

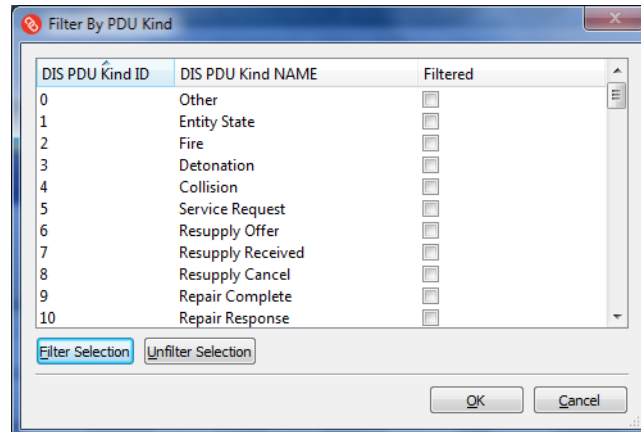


Figure 9-2. Filter By PDU Kind dialog box

2. To change the filtering status of an object, select or clear the check box in the Filtered column for that object.
3. To filter or unfilter multiple objects, select them and click the Filter Selection or Unfilter Selection button.

9.4. Including Custom PDUs in Filter Lists

By default, the Logger lists the PDUs known to VR-Link in the filter lists. If your application uses custom PDUs, you can add them to these lists.

To add custom PDUs to the filter list:

1. Specify a `customPdu` variable in *lgrConfig.xml* for each custom PDU. Each custom PDU must have a unique variable name, as in the following example:

```
<var name="customPDU1" type="list" listtype="string">
  <listitem>MyCustomPdu1</listitem>
  <listitem>186</listitem>
  <listitem>12</listitem>
  <listitem>24</listitem>
</var>

<var name="customPDU2" type="list" listtype="string">
  <listitem>MyCustomPdu2</listitem>
  <listitem>186</listitem>
  <listitem>12</listitem>
  <listitem>24</listitem>
</var>
```

The four fields for each PDU listing are the name of the PDU, the PDU kind, the offset in bytes for the entity ID, and the offset in bytes for the entity type. If there is no entity ID or type, use -1.

2. In the commands section of *lgrConfig.xml*, add each custom PDU to the filter list with an `AddCustomPDU` command, as in the following example:

```
<command domain="Simulation" protocol="DIS" command="AddCustomPDU">
  <param name="pdu" var="customPDU1"/>
</command>
<command domain="Simulation" protocol="DIS" command="AddCustomPDU">
  <param name="pdu" var="customPDU2"/>
</command>
```

9.5. Filtering DIS Messages by Entity ID

In DIS, the Logger can filter by the site:host:entity triplet.



To filter by entity ID, Logger files must use the Logger 4.3 format or later.

To filter messages by entity ID:

1. Open a Logger file.
2. Choose **Filter** → **By Entity ID**. The Filter By Entity ID dialog box opens (Figure 9-3).

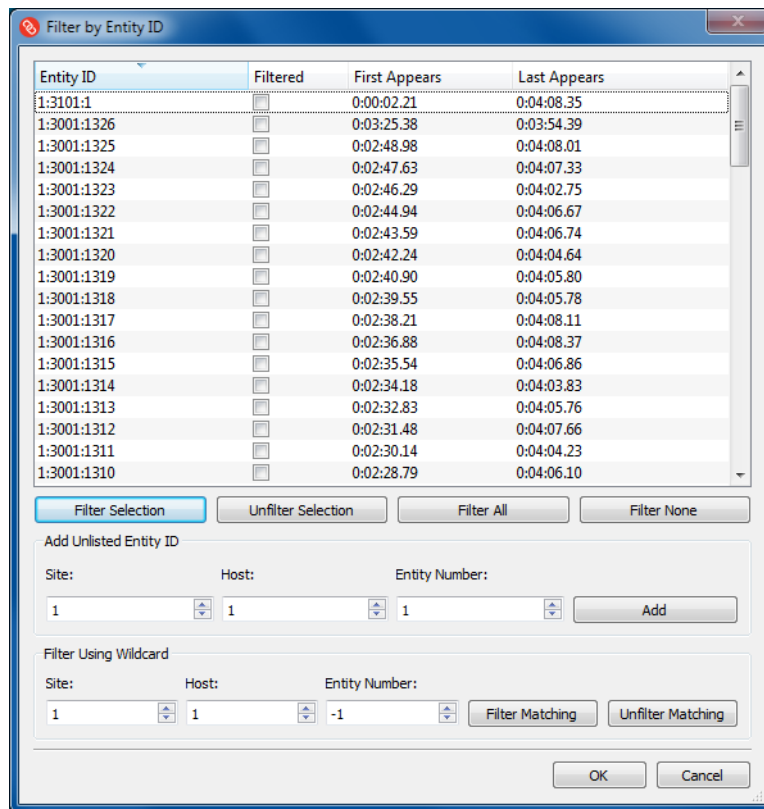


Figure 9-3. Filter by Entity ID dialog box

3. Specify the entity IDs to filter in any of the following ways:
 - To filter or unfilter a single entity ID, select or clear its check box in the Filtered column.
 - To filter or unfilter multiple entity IDs at one time, select the entity IDs using standard click, **CTRL**-click, and **Shift**-click methods, and click Filter Selection or Unfilter Selection. The check boxes in the Filtered column are selected or cleared.
 - To filter or unfilter all entity IDs, click Filter All or Filter None. The check boxes in the Filtered column are selected or cleared.
 - To filter using wildcards, in the Filter Using Wildcard group box, type the values for the site:host:entity triplet or use the spin controls to set them. You can use -1 to represent any value. Click Filter Matching or Unfilter Matching to select or clear the check boxes for all matching entity IDs
4. To add an entity ID to the list:
 - a. In the Add Unlisted Entity ID group box, type the values for the site:host:entity triplet or use the spin controls to set them.
 - b. Click Add. The entity ID is added to the list.
5. Click OK.

9.6. Filtering HLA Messages by Object Name

You can filter HLA messages by object name.



To filter by object name, Logger files must use Logger 4.3 format or later.

To filter messages by object name:

1. Open a Logger file.
2. Choose **Filter** → **By Object Name**. The Filter by Object Name dialog box opens (Figure 9-4).

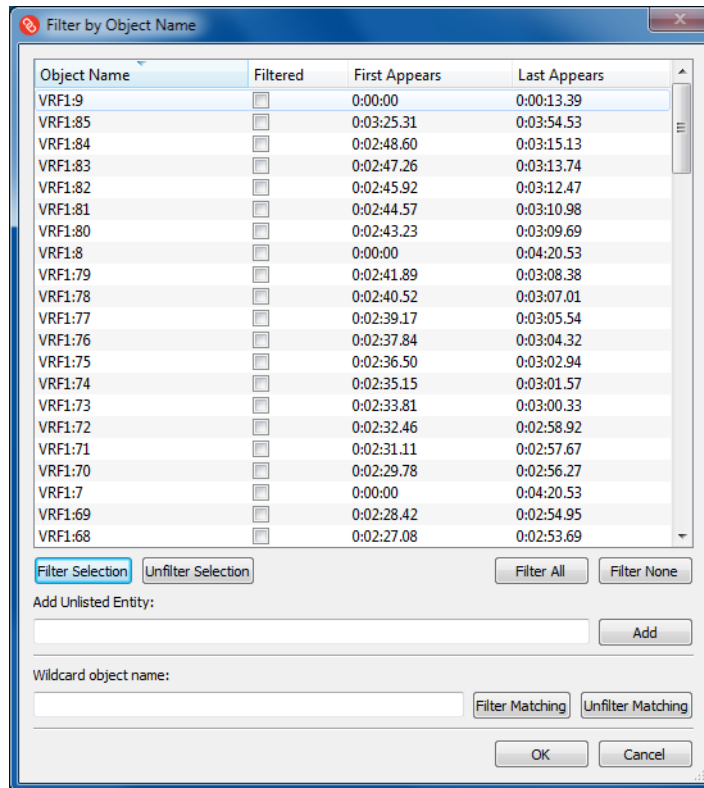


Figure 9-4. Filter by Object Name dialog box

3. Specify the object names to filter in any of the following ways:
 - To filter or unfilter a single object name, select or clear its check box in the Filtered column.
 - To filter or unfilter multiple object names at one time, select the object names using standard click, **CTRL**-click, and **Shift**-click methods, and click Filter Selection or Unfilter Selection. The check boxes in the Filtered column are selected or cleared.
 - To filter or unfilter all object names, click Filter All or Filter None. The check boxes in the Filtered column are selected or cleared.
 - To filter using wildcards, in the Wildcard Object Names box, type a text string that you want to match. Use an asterisk (*) to match an indeterminate number of characters. Use a question mark (?) to match a single character. Click Filter Matching or Unfilter Matching to select or clear the check boxes for all matching object names.
4. To add an object name to the list:
 - a. In the Add Unlisted Entity box, type the object name that you want to add to the list.
 - b. Click Add. The object name is added to the list.

5. Click OK.

9.7. Filtering HLA Exercises by Class

To filter Logger input and output by class:

1. Choose **Filter** → **By Class**. The Filter By Class dialog box opens (Figure 9-5).

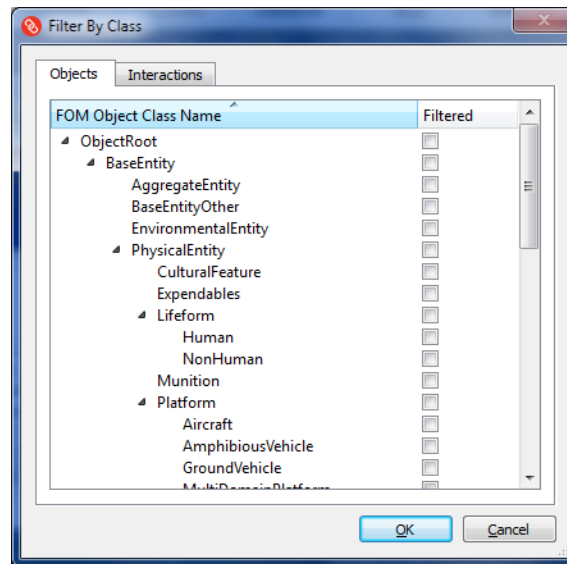


Figure 9-5. Filter By Class dialog box

2. Select the Objects tab to filter object classes. Select the Interactions tab to filter Interaction classes.
3. To change the filtering status of an object, select or clear the check box in the Filtered column for that object.



Selecting or clearing a parent object selects or clears all of its child objects.

9.8. Filtering VR-Forces Messages

If you are recording a simulation that includes messages sent by VR-Forces, MÄK's computer generated forces product, you should filter out the internal messages that VR-Forces sends between the VR-Forces GUI and simulation engine. These messages are meaningless for playback by the Logger. Filtering is enabled by default for both recording and playback.



Filtering of VR-Forces messages is enabled by default. Changing the default message ID range is not recommended unless you have specific instructions from MÄK or a local developer who has changed the range of message IDs used by a customized version of VR-Forces.

To filter out VR-Forces messages:

1. Choose **Filter** → **VR-Forces Messages**. The Filter VR-Forces Messages dialog box opens (Figure 9-6).

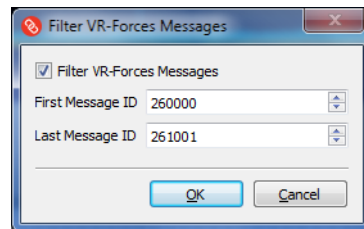


Figure 9-6. Filter VR-Forces Messages dialog box

2. Select the filter VR-Forces Messages check box.
3. Specify the range of message IDs to filter out.
4. Click OK.



10. Analyzing HLA Data

This chapter explains how to use the HLA Traffic Analyzer to analyze HLA data.

The HLA Traffic Analyzer	10-2
Traffic Analyzer Data Views.....	10-2
Hiding the Attributes Pane.....	10-3
How the Traffic Analyzer Evaluates Data.....	10-3
Representing Complex Datatypes.....	10-4
Datatype Representations Included in the Traffic Analyzer	10-4
Viewing Object and Interaction Classes and Attributes.....	10-5
Viewing the Attributes of an Object Class	10-5
Viewing the Parameters of an Interaction Class	10-5
Viewing Object Instances.....	10-6
Viewing Interaction Instances	10-7
Configuring the HLA Traffic Analyzer	10-9
Changing the Data Delimiter	10-10
Mapping Datatypes	10-11
Configuring Attribute Display Options.....	10-11
Mapping Entity Type Enumerations to Strings	10-12
Removing Brackets from the Display of Array Types.....	10-12

10.1. The HLA Traffic Analyzer

The HLA versions of the Logger include a Traffic Analyzer view. The Traffic Analyzer provides details about the FOM you are using and displays the data for the current known object instances and recent interactions. By examining the classes, you can debug problems that you are experiencing in your federation, such as an entity's location not being within the area for the scenario or an interaction missing parameter values.

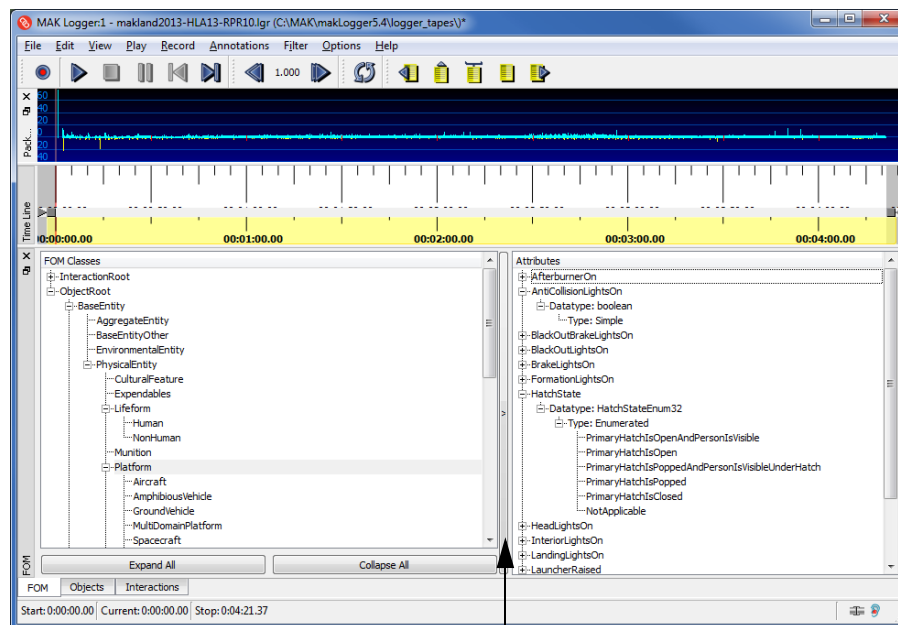
- To display or hide the Traffic Analyzer window, choose **View** → **Traffic Analyzer**.

10.2. Traffic Analyzer Data Views

The traffic analyzer displays HLA data in the following tabs:

- ♦ FOM. Displays the object and interaction classes in the FOM and their attributes.
- ♦ Objects. Displays the Logger's current known objects.
- ♦ Interactions. Displays the recent interactions passing through the Logger pipeline.

Figure 10-1 displays the Traffic Analyzer view.



Toggle Attributes Pane

Figure 10-1. Traffic Analyzer, FOM tab

10.2.1. Hiding the Attributes Pane

Each tab on the Traffic Analyzer has a list of classes in the left window pane and a list of attributes for the selected class in the right window pane. You can hide the Attributes pane.

- To hide or show the Attributes pane, click the Toggle Attributes Pane bar ([Figure 10-1](#)).

10.3. How the Traffic Analyzer Evaluates Data

The encodings for data transmitted in an HLA federation execution are specified by the FOM in the various datatype tables (for example, in HLA 1516, `basicData`, `simpleData`, `fixedRecordData`, and so on). In order to analyze the data and represent it in the display, the HLA Traffic Analyzer must break down the FOM datatypes into basic datatype representations that can be directly interpreted by C++ code. The HLA Traffic Analyzer defines a fixed set of basic datatype representations that it can recognize (listed in “[Datatype Representations Included in the Traffic Analyzer](#),” on page 10-4). For each basic datatype found in the FOM, an association must be established to one of these recognizable datatypes. No association is needed if the FOM uses one of the recognized datatype names (for example, HLA 1.3 FOMs use `float` and `int`, which are directly recognized).

Although there are defined standards for many datatypes, there is no guarantee that a particular datatype has a standard name. Therefore, the Traffic Analyzer has a translation layer that converts datatypes into data representations. These representations are classes that know how to decode a particular datatype.

For example, an object of type ‘`ObjectRoot.Country`’ has an attribute called ‘`Population`’ that contains four bytes of data. The Traffic Analyzer looks up the datatype of the attribute. In this case, the ‘`Population`’ attribute is found to have a datatype “`fixed-number`”. If the Traffic Analyzer knows that `fixed-numbers` are little endian four byte integers, the translation table maps “`fixed-number`” to “`integer-LE`”. The Traffic Analyzer looks up the “`integer-LE`” representation and converts the four bytes of data into an integer. The Traffic Analyzer then displays the value in a readable format.



The Traffic Analyzer gets datatype information from the OMT file for HLA 1.3 and the FED for HLA 1516 and HLA Evolved. For HLA 1.3, the Traffic Analyzer treats the FED file as an OMT file.

Defining translations between datatypes and data representations must be done manually. The Traffic Analyzer defines the most common values, including all of the ones in the RPR FOM. If your federation has a named datatype without a representation, you can edit `<install_dir>/bin/datatypeDefault.mtl` to add your datatype.

The Datatype Mapper lists each of the datatypes in the FOM and its representation. For details, see “[Mapping Datatypes](#),” on page 10-11.

10.3.1. Representing Complex Datatypes

Some attributes are more complex than a single datatype. Arrays, fixed, complex, and other types of attributes are combinations of datatypes. Complex datatypes are decoded separately and then merged using delimiters. For example, a comma-delimited array of numbers looks like this: 5, 4, 3, 2, 1. A delimiter defined as “” merges all values together, like this: 54321.

By default, the Traffic Analyzer uses a comma as the delimiter. You can change the delimiter by editing *datatypeDefault.mtl*, or you can change the value in the Delimiter Mapper (see [“Changing the Data Delimiter,”](#) on page 10-10).

10.3.2. Datatype Representations Included in the Traffic Analyzer

The Traffic Analyzer ships with the following datatype representations, which should cover all but the most proprietary encoding formats:

- int – Signed four byte integer in network or big endian format.
- unsigned-int – Unsigned four byte integer in network or big endian format.
- int-LE – Signed four byte integer in little endian format.
- unsigned-int-LE – Unsigned four byte integer in little endian format.
- long – Signed eight byte integer in network or big endian format.
- unsigned-long – Unsigned eight byte integer in network or big endian format.
- long-LE – Signed eight byte integer in Windows or little endian format.
- unsigned-long-LE – Unsigned eight byte integer in little endian format.
- short – Signed two byte integer in network or big endian format.
- unsigned-short – Unsigned two byte integer in network or big endian format.
- short-LE – Signed two byte integer in little endian format.
- unsigned-short-LE – Unsigned two byte integer in little endian format.
- float – Four byte floating point number in network or big endian format.
- float-LE – Four byte floating point number in little endian format.
- double – Eight byte floating point number in network or big endian format.
- double-LE – Eight byte floating point number in little endian format.
- byte – A single byte written as a number 0-255.
- hex – A single byte written as a hex number 00-FF.
- bool – A single byte representing ‘false’ if zero or ‘true’ if not.
- char – A single byte written as an ASCII value.
- string – A variable number of bytes written as consecutive ASCII values ending in a null.
- unicode – A variable number of bytes written as consecutive “utf16” values ending in a null.

10.4. Viewing Object and Interaction Classes and Attributes

The FOM tab (Figure 10-1) provides a view of all object classes and their attribute information, and all interaction classes and their parameters in the FOM.

The left pane of the FOM window displays an inheritance tree of all known object and interaction classes. The right pane displays the attributes of the selected object or the parameters of the selected interaction. You can expand individual attributes to display datatype information. Complex, fixed, and variant datatypes expand even further to show each one of their components and their datatypes.

The attribute list only shows the attributes specific to the selected object, not its inherited attributes. For example, if you select an object “ObjectRoot.BaseEntity.PhysicalEntity.Platform.Airplane.F18”, the attribute list only shows the F18 attributes, not the inherited Airplane attributes. In order to see the full list for an object you must view all of the parent objects. (In this example, the parent objects would be Airplane, Platform, PhysicalEntity, BaseEntity, and ObjectRoot.)

10.4.1. Viewing the Attributes of an Object Class

- To view an object’s attributes, select the object in the left pane of the FOM tab (Figure 10-1).

10.4.2. Viewing the Parameters of an Interaction Class

- To view an interaction’s parameters, select the interaction in the left pane of the FOM tab (Figure 10-1).

10.5. Viewing Object Instances

The Objects tab (Figure 10-2) displays all objects that are currently known by the Logger. If the Logger is recording, these are the discovered objects; if it is playing, these are the registered objects.

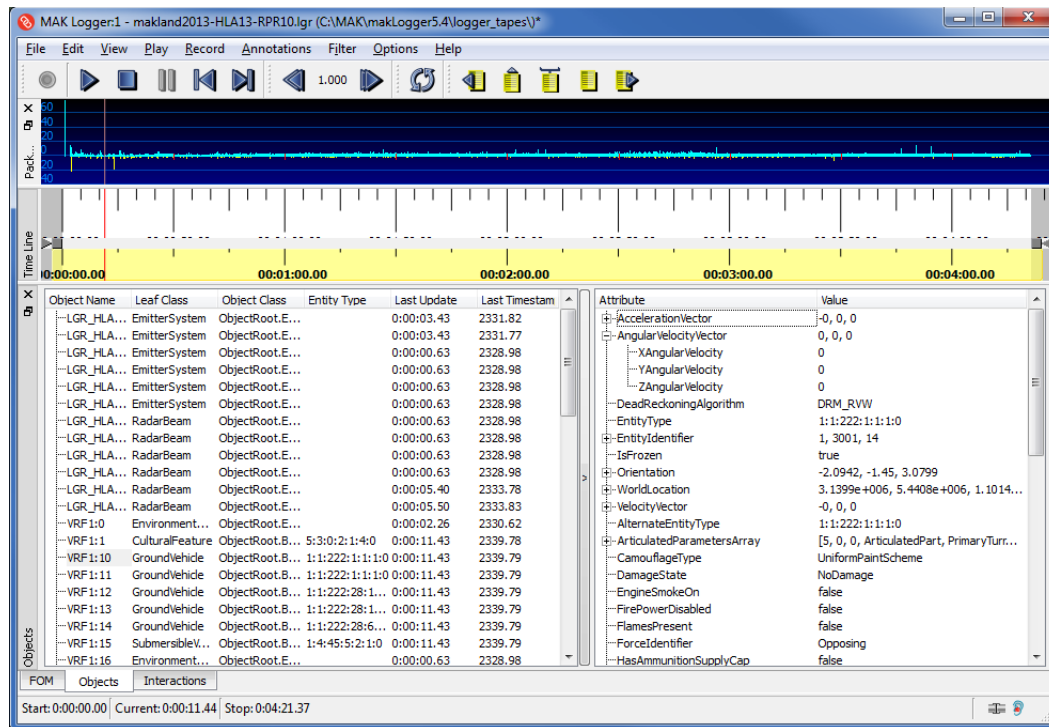


Figure 10-2. Object List

The left pane lists the object name and type of the objects in the federation. If you select an object, attribute information is displayed in the right pane.

The right pane lists all attributes that the selected object publishes. The attribute's values are shown using the correct datatype representation. This way, strings appear as strings, numbers appear as numbers, and all of the data is human readable. Attribute information is displayed as follows:

- If an attribute is a defined enumeration, the enumeration text is shown instead of the number.
- If a datatype is a complex or a fixed value, expanding the attribute displays the individual datatypes.
- If an attribute is an array, the array is divided and each datatype is shown separately.
- If an attribute is a variant, then the correct variant value is shown as long as it is known.

All attribute data is updated in real time.

The attributes displayed on this panel include all inherited attributes.

10.6. Viewing Interaction Instances

The Interactions tab (Figure 10-3) displays the interactions in the HLA exercise and information about them.

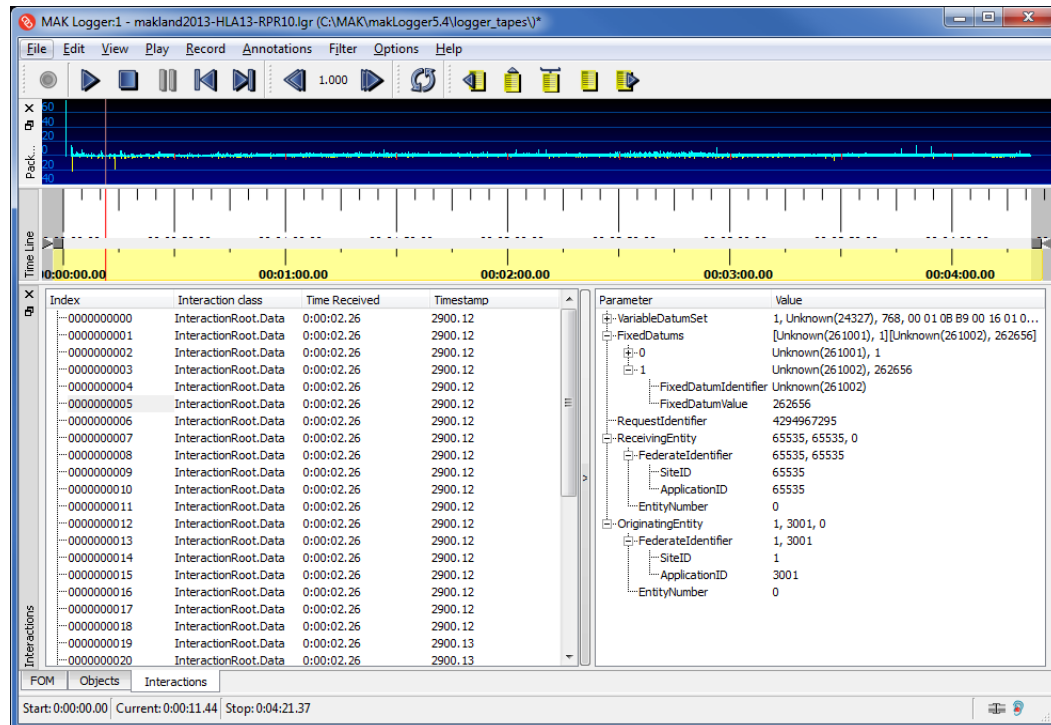


Figure 10-3. Interactions tab

The left pane lists the interactions in the federation. The traffic analyzer lists the 100 most recent interactions sent. As new interactions are added, the oldest ones are dropped from view. You can configure the maximum number of interactions displayed in the HLA Traffic Analyzer Options dialog box or by editing the **max-interaction-count** variable in `<install_dir>/bin/trafficAnalyzerConfig.mtl`.

Each interaction in this panel is listed with its name, its interaction type, and the time that it was received. Selecting an interaction displays information about that particular instance in the right pane.

The right pane lists the parameters for the selected interaction. The parameter values are shown using the correct datatype representation. Strings appear as strings, numbers appear as numbers, and all of the data is human readable. Information is displayed as follows:

- ♦ If a parameter is a defined enumeration, the enumeration text is shown instead of the number.
- ♦ If a datatype is a complex or a fixed value, expanding the attribute displays the individual datatypes.
- ♦ If an parameter is an array, the array is divided and each datatype is shown separately.
- ♦ If an parameter is a variant, the correct variant value is shown as long as it is known.

The parameter details include all inherited parameters.

10.7. Configuring the HLA Traffic Analyzer

You can configure the following Traffic Analyzer settings:

- ♦ Maximum number of interactions. The maximum number of interactions to list in the Interactions List. As new interactions beyond this number are sent or received, the oldest ones are removed.
- ♦ Default enumeration size.
- ♦ Floating point precision.
- ♦ Default delimiter.
- ♦ Datatype delimiters.
- ♦ Datatype mappings
- ♦ Attribute display options.

To edit HLA Traffic Analyzer options:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Traffic Analyzer Settings page ([Figure 10-4](#)).

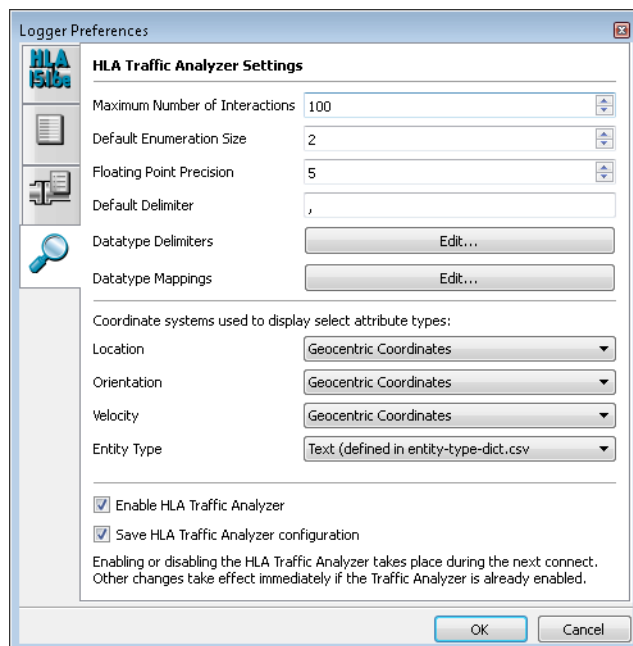


Figure 10-4. Logger Preferences dialog box, HLA Traffic Analyzer Settings

3. Change the values as desired. For details about changing delimiters and datatypes, please see [“Mapping Datatypes,”](#) on page 10-11 and [“Changing the Data Delimiter,”](#) on page 10-10.

10.7.1. Changing the Data Delimiter

As described in “Representing Complex Datatypes,” on page 10-4, the Traffic Analyzer combines the values of complex datatypes using a delimiter. The default delimiter is a comma. You can change the delimiter for complex datatypes, except arrays, using the Delimiter Mapper. (Array elements are always delimited by brackets ([]).)



A complex type “B” found within a complex type “A” uses the delimiter for “B” for its components. The delimiter for “A” is used before and after the “B” instance.

To change the delimiter for a complex datatype:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Traffic Analyzer Settings page (Figure 10-4).
3. On the Datatype Delimiters line, click Edit. The Delimiter Mapper dialog box opens (Figure 10-5).

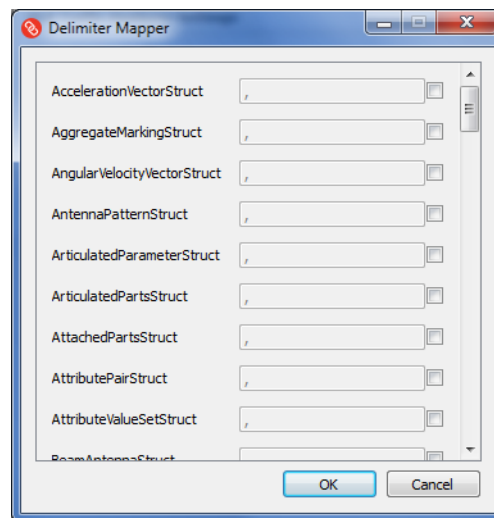


Figure 10-5. Delimiter Mapper dialog box

4. Check the box to the right of the each datatype for which you want to change the delimiter. The text field becomes editable.
5. Replace the current delimiter with the character you want to use.
6. Click OK.

10.7.2. Mapping Datatypes

The Traffic Analyzer lets you change the datatype mappings assigned to known datatypes. Mappings are written to *datatypeDefault.mtl*.

To change datatype mappings:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Traffic Analyzer Settings page (Figure 10-4).
3. On the Datatype Mappings line, click Edit. The Datatype Mapper dialog box opens (Figure 10-6).

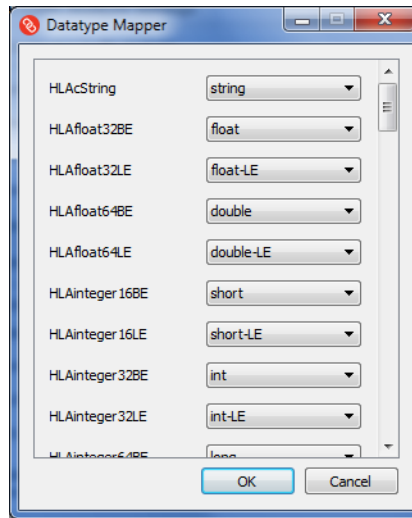


Figure 10-6. Datatype Mapper dialog box

4. For each datatype whose mapping you want to change, select the new datatype from the associated list.
5. Click OK.

10.7.3. Configuring Attribute Display Options

You can specify the coordinate system used to display location, orientation, and velocity for entities. You can also specify how the entity type is displayed.



This option is only available for RPR FOM FOM mappers.

To specify entity display options:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Traffic Analyzer Settings page (Figure 10-4).

3. For each attribute, choose an option from the list.
4. Click OK.

10.7.4. Mapping Entity Type Enumerations to Strings

In the Objects list, the Traffic Analyzer can map entity type enumerations to the strings that the enumerations represent, for example kind, platform, country, and so on. The mapping is done based on either a CSV file or a file using the SISO enumeration XML document format. The Logger provides a file in each format, *./bin/entity-type-dict.csv* and *./bin/OSISO-REF-010.xml*.

Object Name	Leaf Class	Object Class	Entity Type
...VRF 1:38	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:39	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:4	SurfaceVessel	ObjectRoot.BaseEntity	Platform:Surface:225:4:0:0:0
...VRF 1:40	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:41	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:42	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:43	Aircraft	ObjectRoot.BaseEntity	Platform:Air:225:57:1:1:1
...VRF 1:44	Aircraft	ObjectRoot.BaseEntity	Platform:Air:225:57:1:1:1
...VRF 1:45	Aircraft	ObjectRoot.BaseEntity	Platform:Air:225:57:1:1:1
...VRF 1:46	Aircraft	ObjectRoot.BaseEntity	Platform:Air:225:57:1:1:1
...VRF 1:47	CulturalFeature	ObjectRoot.BaseEntity	Cultural feature:Land:0:23:3:0:1
...VRF 1:48	EnvironmentProcess	ObjectRoot.EnvironmentProcess	
...VRF 1:49	SubmersibleVessel	ObjectRoot.BaseEntity	Platform:Subsurface:45:5:2:1:0
...VRF 1:5	Aircraft	ObjectRoot.BaseEntity	Platform:Air:United States:Fighter/Air Defense: General Dyna...
...VRF 1:50	SubmersibleVessel	ObjectRoot.BaseEntity	Platform:Subsurface:45:5:2:1:0
...VRF 1:51	Aircraft	ObjectRoot.BaseEntity	Platform:Air:United States:Antisubmarine Warfare/Patrol: Sikor...
...VRF 1:52	Aircraft	ObjectRoot.BaseEntity	Platform:Air:United States:Fighter/Air Defense: McDonnell-Dou...
...VRF 1:53	Human	ObjectRoot.BaseEntity	Life Form:Land:44:Dismounted Infantry (with):32:1:0
...VRF 1:54	Human	ObjectRoot.BaseEntity	Life Form:Land:44:Dismounted Infantry (with):32:1:0
...VRF 1:55	Human	ObjectRoot.BaseEntity	Life Form:Land:44:Dismounted Infantry (with):32:1:0
...VRF 1:56	CulturalFeature	ObjectRoot.BaseEntity	Cultural feature:Land:Other:POL Storage Tank:0:0:0
...VRF 1:57	GroundVehicle	ObjectRoot.BaseEntity	Platform:Land:22:27:1:0:0
...VRF 1:58	Human	ObjectRoot.BaseEntity	Life Form:Land:44:Dismounted Infantry (with):32:1:0
...VRF 1:6	Aircraft	ObjectRoot.BaseEntity	Platform:Air:United States:Fighter/Air Defense: General Dyna...
...VRF 1:7	Aircraft	ObjectRoot.BaseEntity	Platform:Air:United States:Bomber: Northrop B-2:0:0
...VRF 1:8	EnvironmentProcess	ObjectRoot.EnvironmentProcess	

Figure 10-7. Entity type mappings

The file to be used is specified in *trafficAnalyzerConfig.mtl* with the `entity-type-map-file` parameter. The default is *OSISO-REF-010.xml*. The syntax is:

```
(entity-type-map-file "OSISO-REF-010.xml")
```

10.7.5. Removing Brackets from the Display of Array Types

By default, the display of array elements in the Traffic Analyzer uses brackets to delimit each element. You can remove the brackets from any array type (Figure 10-8). For example, in the case articulated parts, if you look in the *MAK-RPR1-1-1.xml* file you will see it is of type `ArticulatedParameterStructArray`, which is composed of elements of type `ArticulatedParameterStruct`. If you add an entry like the following, it removes the brackets from the display of this array data.

```
(arrayBracketMap "ArticulatedParameterStruct" 0)
```

ArticulatedParametersArray	[0, 3, 0, ArticulatedPart, PrimaryTurretNumber1, Azimuth, 0][0, 53, 0, ArticulatedPart, PrimaryTurretNumber1, AzimuthRate, 0]
0	0, 3, 0, ArticulatedPart, PrimaryTurretNumber1, Azimuth, 0
1	0, 53, 0, ArticulatedPart, PrimaryTurretNumber1, AzimuthRate, 0
2	0, 0, 1, ArticulatedPart, PrimaryTurretNumber2, Elevation, 0.785
3	0, 0, 1, ArticulatedPart, PrimaryTurretNumber2, ElevationRate, 0

ArticulatedParametersArray	0, 0, 0, ArticulatedPart, PrimaryTurretNumber1, Azimuth, 00, 118, 0, ArticulatedPart, PrimaryTurretNumber1, AzimuthRate, 00, 118
0	0, 0, 0, ArticulatedPart, PrimaryTurretNumber1, Azimuth, 0
1	0, 118, 0, ArticulatedPart, PrimaryTurretNumber1, AzimuthRate, 0
2	0, 118, 1, ArticulatedPart, PrimaryTurretNumber2, Elevation, 0.785
3	0, 119, 1, ArticulatedPart, PrimaryTurretNumber2, ElevationRate, 0
4	0, 120, 2, ArticulatedPart, PrimaryLauncher1, Azimuth, 0
5	0, 120, 2, ArticulatedPart, PrimaryLauncher1, AzimuthRate, 0

Figure 10-8. Array data, with and without brackets

In almost all cases, array data is more readable with brackets. The one case where it makes sense to not use brackets is when the data is an array of characters.

Attributes that are ASCII strings (such as RPR FOM marking text) are sometimes encoded as an array of characters or other datatype (for example, HLAoctet). For strings, the brackets add unnecessary clutter. The brackets can be removed using the **arrayBracketMap** option in the *datatypeDefault.mtl* file, but that would affect all arrays of the given data type. As an alternative, you can map a FOM attribute name or data type field name to another datatype (for example, string) using the **attributeMap** option. This option only affects attributes or data type fields with the given name, for example:

```
(attributeMap "MarkingData" "string")
```




11. Editing Logger Files

This chapter explains how to merge, concatenate, and edit Logger files.

Editing Files.....	11-2
Reverting to the Saved Version of a File	11-2
Clearing and Deleting Segments of a File.....	11-3
Merging Files	11-5
Concatenating Files	11-5
Trimming Files	11-6
Offsetting Files	11-6

11.1. Editing Files

The Logger has the following options for editing files:

- ♦ Merge two files.
- ♦ Add a file to the beginning or end of another file.
- ♦ Clear a portion of a file of data and leave empty space.
- ♦ Delete a portion of a file.
- ♦ Trim dead space at the beginning of a file.
- ♦ Offset the timestamps of packets.



When you edit a file, the changes are not saved automatically. The title bar adds an asterisk to the file name to indicate that it has changed. To retain your edits, you must save the edited file when you are finishing editing it.

11.1.1. Reverting to the Saved Version of a File

If you edit a file and decide that you want to remove all edits, you can revert to the previously saved version of the file. This option is available only for files that have a saved version. You cannot apply it to unsaved, new recordings.

- To revert to the previously saved version of a file, choose **File** → **Revert**.

11.2. Clearing and Deleting Segments of a File

You can select a portion of a file and clear it or delete it. Clearing a portion of a file removes all packets within the selection, but does not change the timestamps of the packets that follow the cleared section. The result is empty space in the file.

If you delete a section of a file, all packets and any associated annotations are removed. The packets that come after the deleted section are offset to fill in the empty space. If a range annotation spans the deleted section its start point or end point is offset to the appropriate amount of time.

Figure 11-1 shows a logger file with the segment from 0:30 to 1:30 selected for editing. If the segment is cleared, the file stays the same length, but there are no messages in the cleared segment. If the segment is deleted, the file is shortened by the amount of time deleted. There is no dead space in the file.



Remember, when you clear or delete a portion of a Logger file, it is the white segment in the time line that gets edited. The gray segments are preserved.

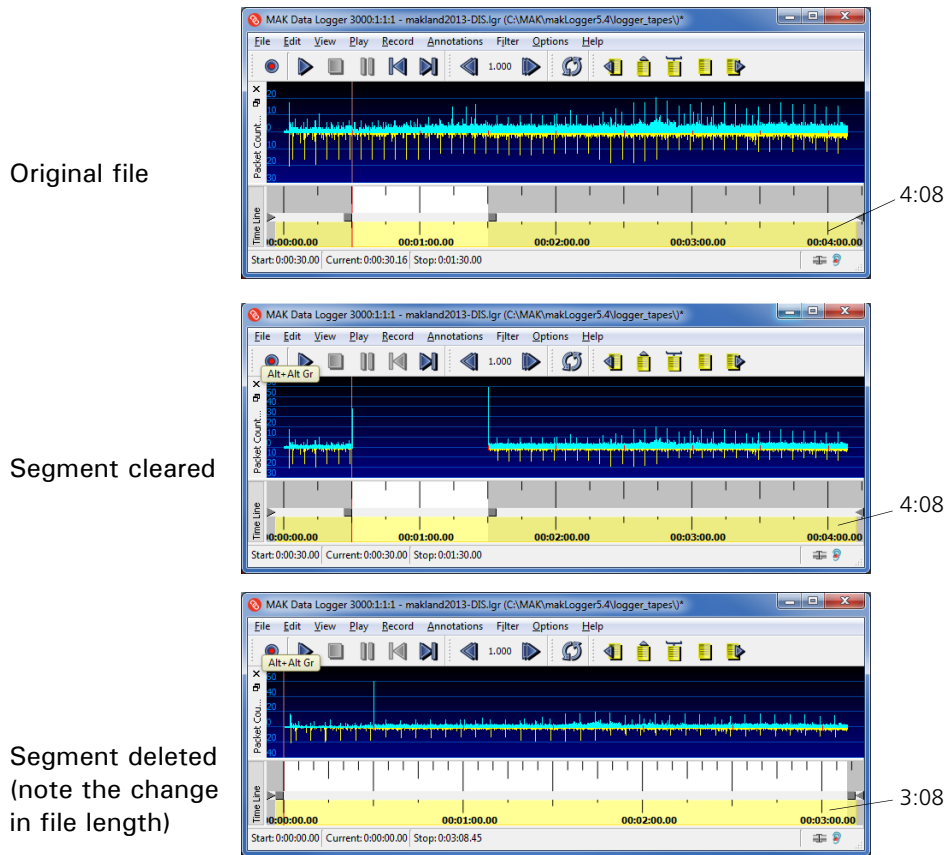


Figure 11-1. Clearing and deleting file segments

The Logger adds data, if necessary, to properly remove or add objects that are affected by a clear or delete operation.



When you delete a portion of a file, it does not get placed on the clipboard. You cannot paste it back into the file.

To clear a portion of a file:

1. Open the file.
2. Use the timeline sliders to select the segment you want to clear.
3. Choose **Edit** → **Clear Section**.

To delete a portion of a file:

1. Open the file.
2. Use the timeline sliders to select the segment you want to delete.

3. Choose **Edit** → **Delete Section**.
4. Save the edited file.

11.3. Merging Files

When you merge files the packets for the file that you are merging in are added to the original file in order of the timestamps. The Logger does not do any checking for duplicate names. It adds new checkpoints using the current checkpoint settings.



To successfully merge HLA files, they must have been created with the same FOM.

To merge two files:

1. Open a file.
2. Choose **Edit** → **Merge In File**. An Open dialog box opens.
3. Select the file that you want to merge in.
4. Click Open. The selected file is merged into the first file.
5. Save the edited file.

11.4. Concatenating Files

You can concatenate (append) a file to the beginning or end of another file.



To successfully concatenate HLA files, they must have been created with the same FOM.

To concatenate files:

1. Open a file.
2. Choose **Edit** → **Append File to End** or **Prefix File to Start**. An Open dialog box opens.
3. Select the file that you want to concatenate.
4. Click Open. The selected file is concatenated to the open file.
5. Save the edited file.

11.5. Trimming Files

When you trim a file, empty space at the beginning is removed and packets are offset so that the first packet is sent at the beginning of the file.

To trim a file:

1. Open the file.
2. Choose **Edit** → **Trim Start**.
3. Save the edited file.

11.6. Offsetting Files

When you offset a file, a specified number of seconds is added to the time stamp of each packet. (This results in dead space at the start of the file.)

To offset a file:

1. Open the file.
2. Choose **Edit** → **Offset Time**. The Set Offset dialog box opens.
3. Type the time, in seconds, that you want to offset the file.
4. Click OK.
5. Save the edited file.



12. Exporting Logger Data

This chapter explains how to export Logger files to various data formats.

Exporting Data	12-2
Exporting Data to a Database	12-3
Configuring an ODBC Data Source	12-6
Exporting DIS Data to a Database	12-7
Exporting HLA Data to a Database.....	12-8
The Export Log.....	12-8
Configuring a Data Source for Microsoft Access	12-9
Installing MySQL and Configuring a Data Source.....	12-13
Installing MySQL and the MySQL ODBC Driver	12-13
Creating a MySQL Database.....	12-14
Configuring a Data Source for MySQL.....	12-14
Importing Logger Data into a Spreadsheet.....	12-17
Importing Data from MySQL to Microsoft Excel	12-17
Importing Data from Access to Excel	12-21

12.1. Exporting Data

You can export some or all of the data in a Logger file to another supported format.

- ♦ You cannot export data while you are in Record mode.
- ♦ Before you can export data to a database, you must set up a data source. For details, please see [“Configuring an ODBC Data Source,”](#) on page 12-6, [“Configuring a Data Source for Microsoft Access,”](#) on page 12-9, and [“Installing MySQL and Configuring a Data Source,”](#) on page 12-13.
- ♦ The Logger does not apply filters when you export data.
- ♦ When you export data, you can have the data resampled with the current check-point policy. For more information, please see [“Configuring Indexing,”](#) on page 5-3.

To export data:

1. Open a Logger file.
2. Choose **File** → **Export to** → *format*, where *format* is a supported format, such as Plain Text. A dialog box appropriate to the export type opens.
3. To export to another Logger file or to text, type a name for the new file and click OK. For other export types, please see the additional procedures in this chapter.

To export part of a Logger file:

1. Use the timeline sliders to select the portion of a Logger file that you want to export.
2. Choose **File** → **Export Selection To** → *format*.
3. To export to another Logger file or to text, type a name for the new file and click OK. For other export types, please see the additional procedures in this chapter.

12.2. Exporting Data to a Database

The Logger can export data to a database using ODBC. Please see *MÄK Data Logger Release Notes* for a list of ODBC drivers supported. The Logger creates tables that contain the messages that it can decode. Optionally, you can create an object instance table and a checkpoint information table. You can apply filters to the data when it is exported.

The Logger uses underscores to separate words used as table and row names. Each PDU or HLA class has its own table. Variable length data is stored as indexed data in a subtable that contains the parent table's name followed by the variable's name. For example, if the parent table is aggregateentity, the subtable might be aggregateentity_silentaggregates.

Notes

- Before you can export data to a database, you must set up a data source. For details, please see [“Configuring an ODBC Data Source,”](#) on page 12-6, [“Configuring a Data Source for Microsoft Access,”](#) on page 12-9, and [“Installing MySQL and Configuring a Data Source,”](#) on page 12-13.
- The 32 bit version of the Logger can only export to data sources created by 32 bit versions of Microsoft Access or MySQL. Similarly, the 64 bit version of the Logger can only export to data sources created with a 64 bit version of Microsoft Access or MySQL.
- You do not have to purchase Microsoft Access to set up a data source for this format. For details, please see [“Configuring a Data Source for Microsoft Access,”](#) on page 12-9.
- When the Logger exports data to an existing database, it only overwrites tables for which it is exporting data. If the database has existing tables for which there is no equivalent in the simulation being exported, the resulting database will contain data that does not represent any actual simulation. For example if you export a simulation that contains detonation interactions to database myLogger and then you export a simulation that does not have detonation interactions to myLogger, the detonation interaction data from the first simulation will still be in the database.

It is your responsibility to ensure that data is exported to a clean database.

- The Logger does not have to be connected to an exercise to export data. However if you use a custom filter, be sure that the filter does not require the Logger to be connected.

To export data to a database:

1. Open the file you want to export or stop a current recording and enter playback mode.
2. Optionally, select a segment of the file using the timeline sliders.

3. Choose **File** → **Export to** → **Database** or **File** → **Export Selection To** → **Database**.
The ODBC Export Settings dialog box opens (Figure 12-1 and Figure 12-2).

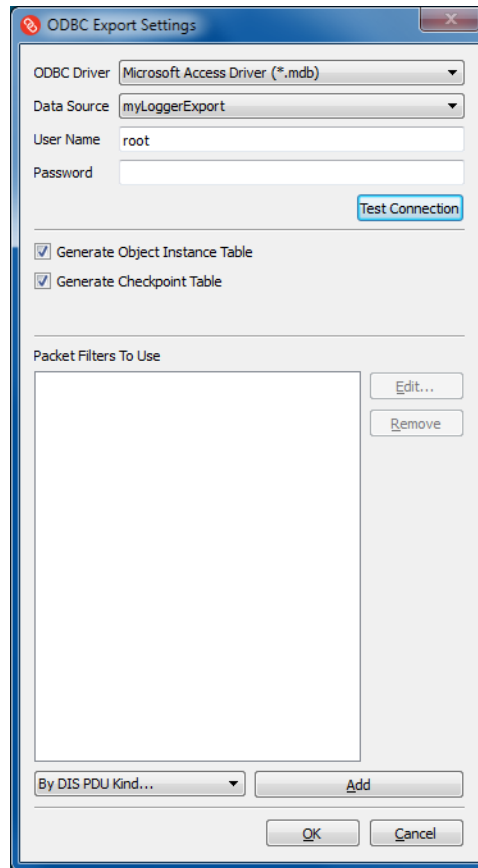


Figure 12-1. ODBC Export Settings dialog box (DIS)

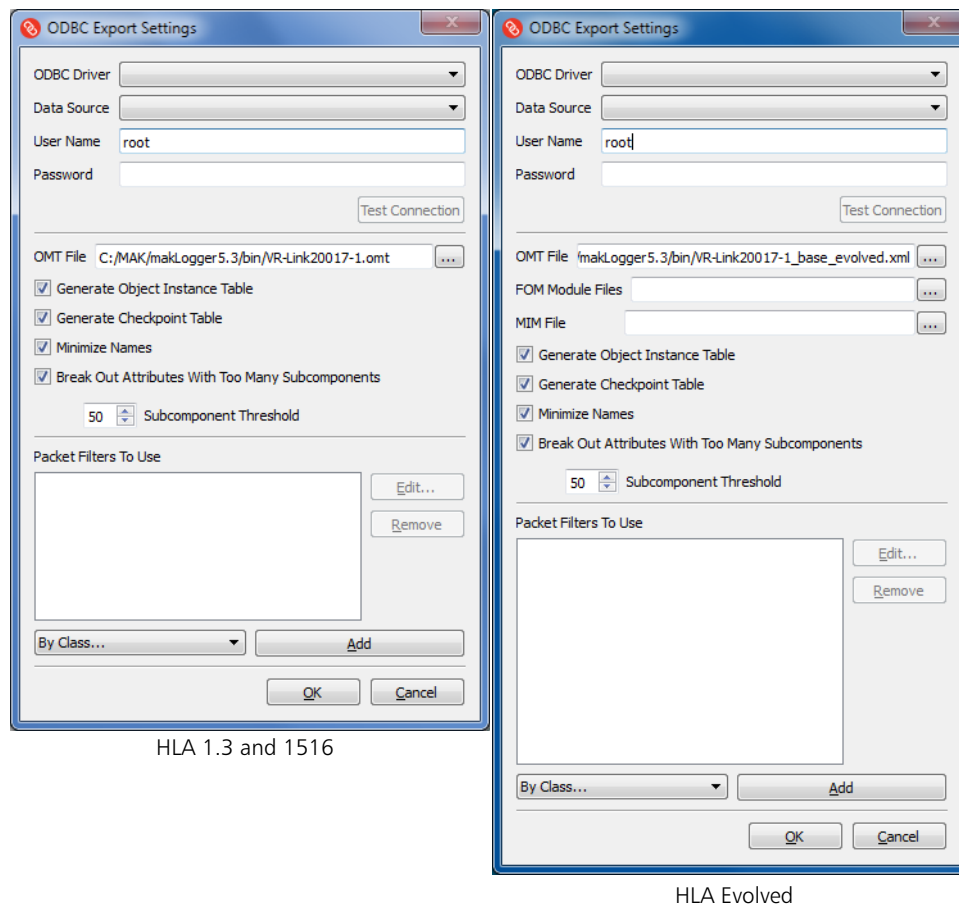


Figure 12-2. ODBC Export Settings dialog box (HLA)

4. Select an ODBC driver in the ODBC Driver list.
5. Select a data source from the Data Source list.
6. For export to MySQL, enter the user name and password for the data source.
7. Optionally, click Test Connection to test the connection to the data source.
8. For HLA Evolved only, optionally add FOM Modules and a MIM file.

The FOM Module Files entry is a list of files separated by semi-colons. The list can be modified directly or populated using the FOM Module browse button. It selects a single file but it is additive. Each time it is used, the selected file is added to the contents of the FOM Modules entry. If existing values need to be removed, you must edit them directly.

9. Optionally, check or clear the check boxes for the optional tables.

10. Optionally, for HLA export, change the threshold for breaking up attributes into subtables or disable this feature. (RPR FOM 1.0 files should not need to do this, but RPR FOM 2.0 files do. For more information, please see [“Exporting HLA Data to a Database,”](#) on page 12-8.)
11. Optionally, specify a filter to use while exporting data, as follows:
 - a. Select a filter type from the drop-down list.
 - b. Click Add.
 - c. Select the filter in the Packet Filters to Use list.
 - d. Click Edit.
 - e. Edit the list of classes to filter. For details about editing filter lists, please see [“Filtering Logger Input and Output,”](#) on page 9-2.
12. Click OK.

12.2.1. Configuring an ODBC Data Source

You must set up a data source before you open the Export To Database dialog box.

On Linux systems you must edit configuration files. GUIs vary depending on the operating system. The Logger uses the iODBC library to provide ODBC capability. Please see the following web site for details about configuring ODBC:

<http://www.iodbc.org>

In Windows you create and configure data sources using the ODBC Data Source Administrator, as described later in this chapter.



The 32 bit version of the Logger can only export to data sources created by 32 bit versions of Microsoft Access or MySQL. Similarly, the 64 bit version of the Logger can only export to data sources created with a 64 bit version of Microsoft Access or MySQL.

12.2.2. Exporting DIS Data to a Database

The Logger supports export to database of the following DIS PDUs:

- ♦ Aggregate State
- ♦ Collision
- ♦ Designator
- ♦ Detonation
- ♦ Electronic Emission
- ♦ Entity State
- ♦ Fire
- ♦ Receiver
- ♦ Signal
- ♦ Start
- ♦ Stop
- ♦ Transmitter
- ♦ Unknown.

You can use the plug-in API to write additional PDU encoders, if needed.

12.2.3. Exporting HLA Data to a Database

The Logger can export data from any FOM. If you are using HLA Evolved, it can export data from any FOM module. This is done by processing the FOM's OMT or XML file. One of these files is required to export any HLA data. If the OMT or XML file is poorly formed, incomplete, or ambiguous, or if a data format does not map conceptually to a database table/row format, the Logger might not be able to decode all data. However, the generic implementation handles the majority of data.

The Logger supports export of the following HLA data:

- ♦ Simple Data Types.
- ♦ Complex Data Types.
- ♦ Array Data (XML FOMs only).
- ♦ Fixed Cardinality Data.

There is no defined HLA database schema. The schema is dynamically generated using the OMT file. HLA class names can be quite long. The Logger has an option to use the shortest unique name possible. For example, instead of using `HLAObject_PhysicalObject_Platform_Aircraft` it can use `Aircraft`. If there is name collision, each immediate parent class is prefixed to the name until a unique name is found.

When exporting to an Access database, there is a limit of 255 fields for any given table. It is possible that the number of fields generated for a class will exceed this limit. As an example, the RPR FOM 2.0017 `BaseEntity.PhysicalEntity.Platform` subclasses generate over 400 fields, not including the variable length attributes that are already broken out into subtables. The Logger can export these classes by breaking out additional attributes and parameters into subtables. It uses the number of fields an attribute will generate (based on the data type subcomponents) as a trigger. When the number of subcomponents exceeds a specified threshold, the Logger can put that attribute into a separate subtable similar to the variable length data attributes. You can configure whether this threshold is considered and the number of subcomponents it takes in the ODBC Export Settings dialog box.

12.2.4. The Export Log

When you export Logger data, the Logger writes information about the export process to `sql_export.txt` in the `./bin` directory from which you are running the Logger. The log file is important because it identifies the data that cannot be exported.

12.3. Configuring a Data Source for Microsoft Access

If you want to export data to Microsoft Access, you must configure a data source. You do not have to have Access installed to complete this procedure. However, you must have a Microsoft Access driver installed. Installing Microsoft Office provides this driver. If you are running a 64-bit Logger, you must install the 64-bit version of MS Office. If you are running the 32-bit Logger (even on a 64-bit computer, you must install the 32-bit version of Office. (You will not be able to manipulate the database if Access is not installed. However, you can open the database in Microsoft Excel, as described in [“Importing Data from Access to Excel,”](#) on page 12-21.)



The 32-bit version of the Logger can only export to data sources created by 32-bit versions of Microsoft Access or MySQL. Similarly, the 64-bit version of the Logger can only export to data sources created with a 64-bit version of Microsoft Access or MySQL.

To configure an ODBC Data Source for Microsoft Access:

1. Open the ODBC Data Source Administrator dialog box. The procedure varies depending on your operating system and version of the Logger, as follows:

If you are running 64-bit Logger on Windows 7:

- a. Open the Control Panel.
- b. Select System and Security.
- c. Select Administrative Tools.
- d. Select Data Sources (ODBC).

If you are running 32-bit Logger on Windows 7 64-bit, run
`C:\Windows\SysWOW64\odbcad32.exe`.

The ODBC Data Source Administrator dialog box opens ([Figure 12-3](#)).

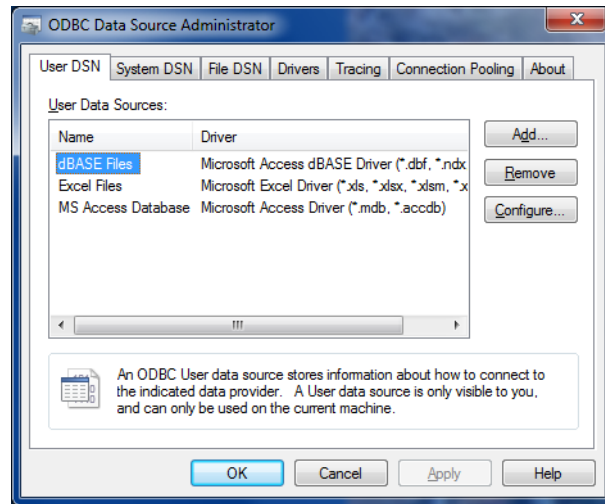


Figure 12-3. ODBC Data Sources Administrator

2. If you want the data source to be available to all users, select the System DSN tab. If you want it to be available just to yourself, select the User DSN tab.
3. Click Add. The Create New Data Source window opens (Figure 12-4).

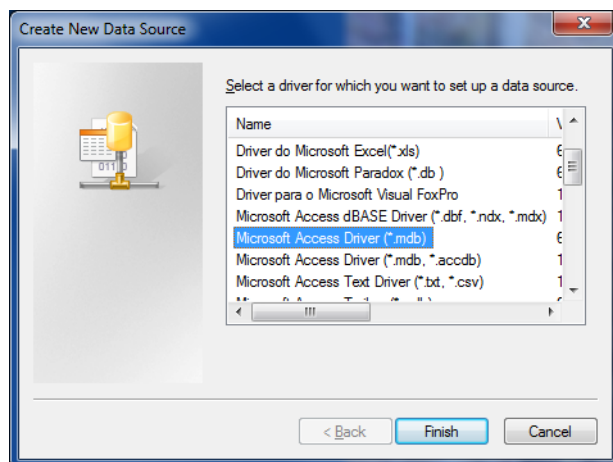


Figure 12-4. Create New Data Source window

4. Select Microsoft Access Driver (*.mdb). (The Logger looks for a driver with this exact text. Any other driver will not be recognized.)
5. Click Finish. The ODBC Microsoft Access Setup dialog box opens (Figure 12-5).

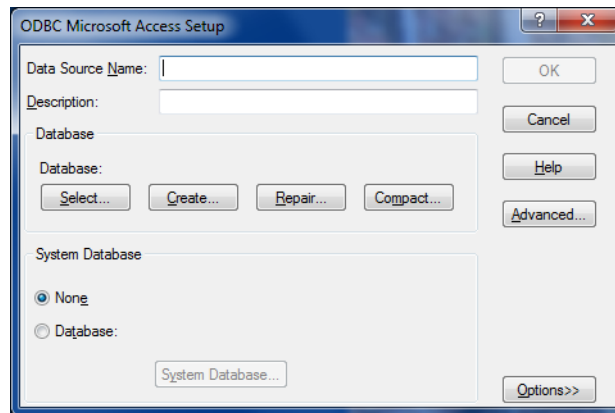


Figure 12-5. ODBC Microsoft Access Setup dialog box

6. In the Data Source Name box, type a name for the data source.
7. In the Database group box, click Create. The New Database dialog box opens (Figure 12-6).

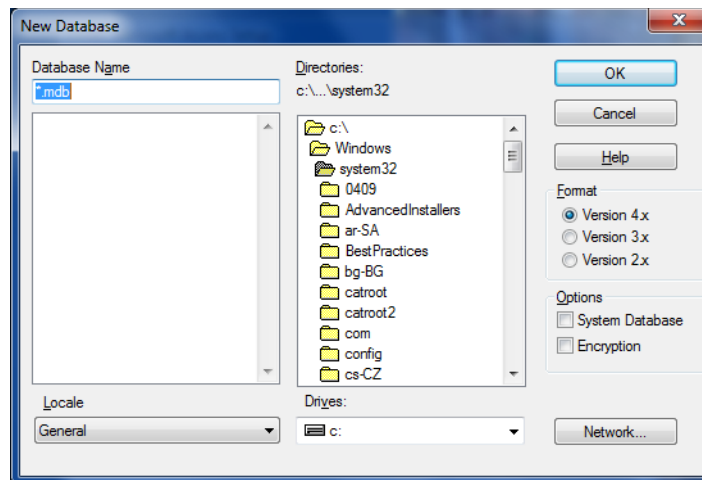


Figure 12-6. New Database dialog box

8. In the Database Name box, type a name for the database, for example, *myNewLoggerExport.mdb*.
9. In the Directories list, select the directory where you want to create the database (Figure 12-7).

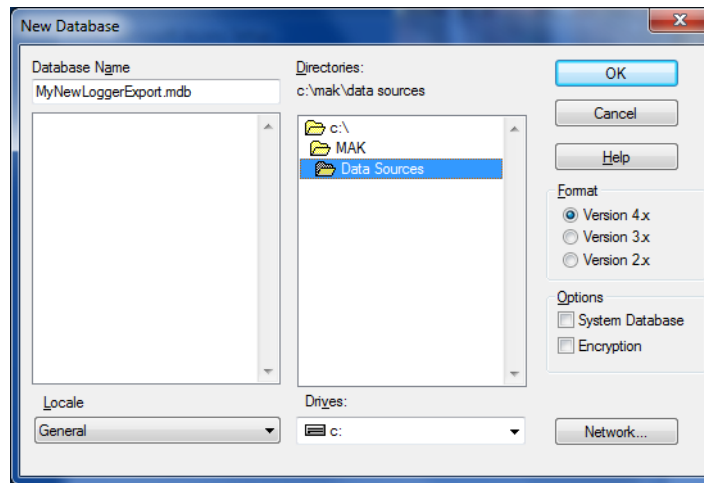


Figure 12-7. Completed New Database configuration

10. Click OK. An informational prompt advises you that the database has been created (Figure 12-8).

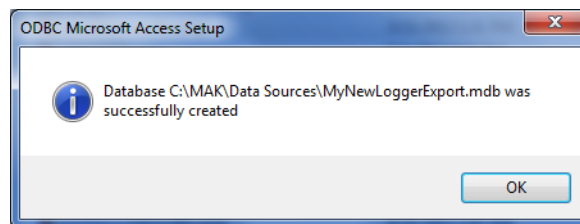


Figure 12-8. Information prompt

11. Click OK.
12. Continue clicking OK to back out of the dialog box sequence.

The next time you export a Logger file to database using the Microsoft Access driver and data source, it is exported to the database file you created in this procedure.

12.4. Installing MySQL and Configuring a Data Source

This section explains how to install MySQL on Windows and configure a data source. Before you can configure a data source, you must install MySQL and the MySQL ODBC driver and you must create a MySQL database to configure as the data source. The procedure in this section was developed using MySQL 5.1. Please see *MÄK Data Logger Release Notes* for a list of supported versions of MySQL. You must use a supported version.



The 32 bit version of the Logger can only export to data sources created by 32 bit versions of Microsoft Access or MySQL. Similarly, the 64 bit version of the Logger can only export to data sources created with a 64 bit version of Microsoft Access or MySQL.

12.4.1. Installing MySQL and the MySQL ODBC Driver

When you install MySQL, you need to set the root password. Then you need to create a database to which you can export Logger data.

To install MySQL and the MySQL ODBC driver:

1. Run the installer for the ODBC driver (called the *mysql-connector-odbc-<version>.msi*).
2. Run the installer for MySQL. Follow the prompts in the installation wizard. When the installation wizard is complete, it prompts you to configure MySQL Server.
3. Select the Configure the MySQL Server Now check box and click Finish. The MySQL Server Instance Configuration Wizard starts.
4. When you are prompted to configure an instance, choose Configure Instance or Reconfigure Instance, depending on which option is available. Click Next.
5. Choose Standard Configuration. Click Next.
6. Choose Install as Windows Service and Include Bin Directory in Windows PATH. Click Next.
7. On the Security Settings page, enter a new root password. If your system has a root password and you know what it is, enter it as the current root password. Most Windows systems probably do not have a root password, so you can leave this field blank. Optionally, create an Anonymous account. Click Next.
8. On the next screen, click Execute.

12.4.2. Creating a MySQL Database

You must create a database before you can set up a data source.

To create a MySQL database:

1. Choose Programs → MySQL → MySQL Server 5.1 → MySQL Command Line Client. A command window opens and you are prompted to enter a password. (Alternatively, open a command window, cd to the MySQL *bin* directory and enter `mysql -u root -p`.)
2. Type the MySQL password.
3. Type:

```
create database LoggerExport;
```

This creates a database called LoggerExport (or whatever name you choose to use).
4. Type:

```
exit
```

12.4.3. Configuring a Data Source for MySQL

The Logger database export feature supports the following MySQL ODBC drivers:

- ♦ MySQL ODBC 2.50 Driver
- ♦ MySQL ODBC 5.351 Driver
- ♦ MySQL ODBC 5.1 Driver
- ♦ MySQL ODBC 5.2w Driver.

For the VC10 64-bit Logger, a 64 bit connector is required.

The MySQL 5.1 server and connector can be downloaded at:

- ♦ server: <http://dev.mysql.com/downloads/mirror.php?id=411970>
- ♦ connector: <http://dev.mysql.com/downloads/mirror.php?id=411736>

Please see *MÄK Data Logger Release Notes* for updates to this list.

To configure an ODBC data source for MySQL:

1. Open the ODBC Data Source Administrator dialog box. The procedure varies depending on your operating system and version of the Logger, as follows:

If you are running 64-bit Logger on Windows 7:

- a. Open the Control Panel.
- b. Select System and Security.
- c. Select Administrative Tools.
- d. Select Data Sources (ODBC).

If you are running 32-bit Logger on Windows 7 64-bit, run
C:\Windows\SysWOW64\odbcad32.exe.

The ODBC Data Source Administrator dialog box opens ([Figure 12-9](#)).

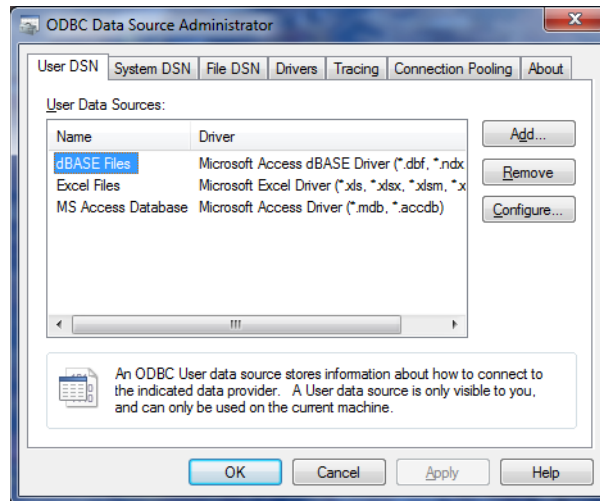


Figure 12-9. ODBC Data Source Administrator dialog box

2. Click Add. The Create New Data Source dialog box opens ([Figure 12-10](#)).

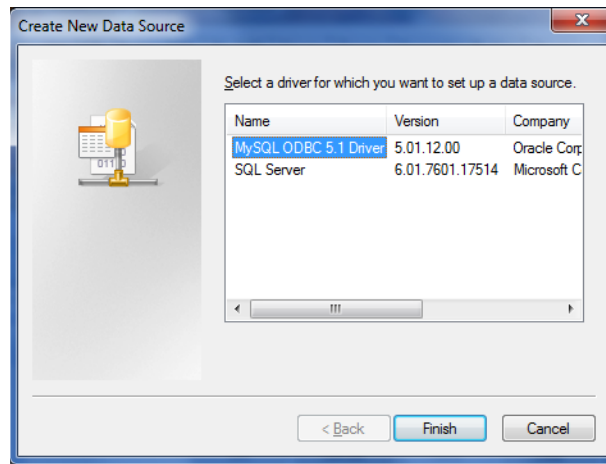


Figure 12-10. Create New Data Source dialog box

3. Select the MySQL ODBC driver.
4. Click Finish. The ODBC Data Source Configuration dialog box opens.

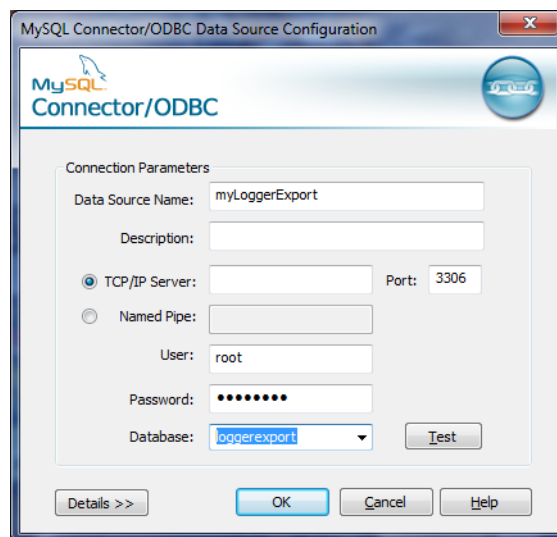


Figure 12-11. ODBC Data Source Configuration dialog box

5. Type a name for your data source.
6. Complete the Server, User, and Password fields.
7. Enter the name of the database that you created for Logger data.
8. Click OK. The new data source is displayed in the ODBC Data Source Administrator dialog box (Figure 12-12).

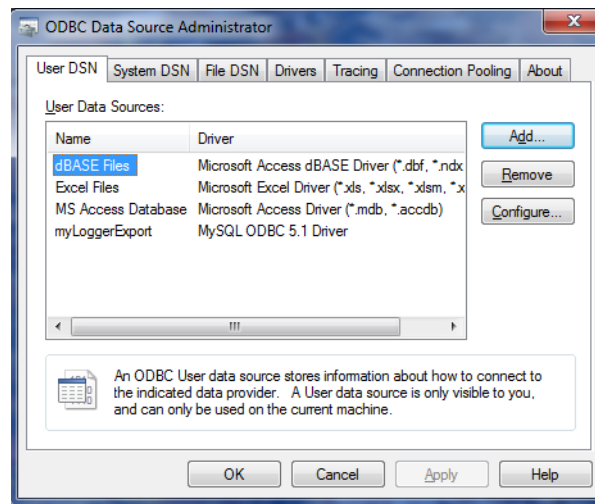


Figure 12-12. ODBC Data Source Administrator dialog box

12.5. Importing Logger Data into a Spreadsheet

After you export Logger data to a database, you can examine and manipulate the data just as you would any database. One common way to view the data is to import it into a spreadsheet and use the spreadsheet's tools to analyze it, graph it, and so on.

12.5.1. Importing Data from MySQL to Microsoft Excel

This section explains how to import data stored in a MySQL database into Microsoft Excel. (This procedure uses Microsoft Excel 2010. Other versions may have slightly different menu options and dialog box names.)

To import data from a MySQL database to Microsoft Excel:

1. Start Excel.
2. Choose **Data** → **From Other Sources** → **From Microsoft Query**. The Choose Data Source dialog box opens (Figure 12-13).

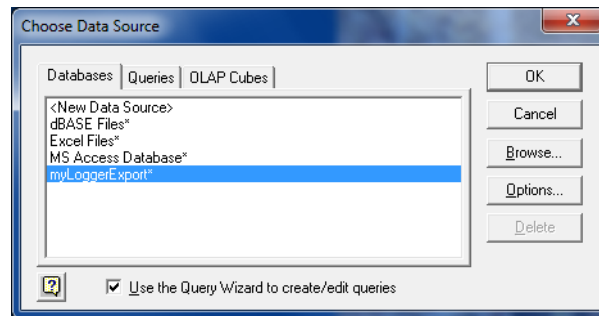


Figure 12-13. Choose Data Source dialog box

3. In the window, select the data source from which you want to import data.
4. Click OK. The Query Wizard - Choose Columns dialog box opens (Figure 12-14). In the Available Tables and Columns window is a list of tables in the database. You can expand each table for a list of the columns in the table.

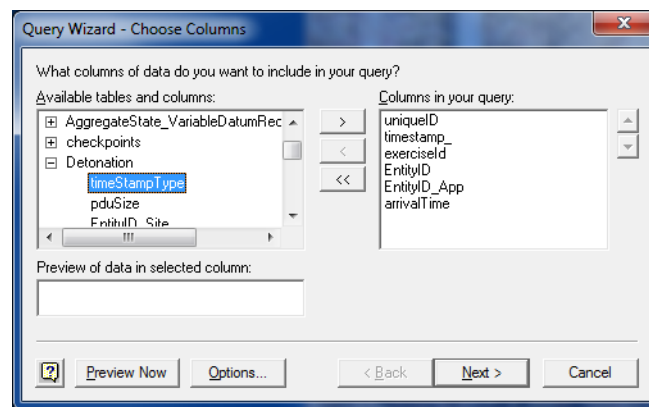


Figure 12-14. Query Wizard - Choose Columns

5. Select the tables and columns that you want to import to Excel.
6. Click the right-facing arrow to move the selected tables and columns to the Columns In Your Query window.
7. Click Next. The Query Wizard - Filter Data dialog box opens (Figure 12-15).

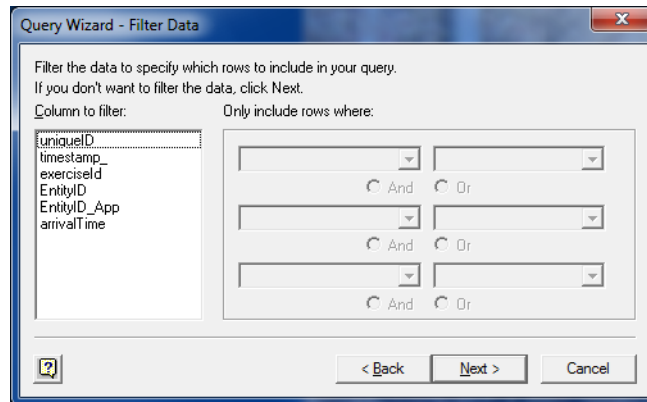


Figure 12-15. Query Wizard - Filter Data

8. Set up any filtering that you want, then click Next. The Query Wizard - Sort Order dialog box opens (Figure 12-16).

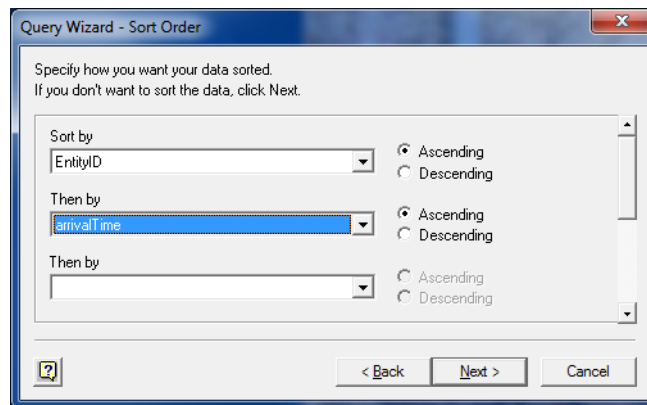


Figure 12-16. Query Wizard - Sort Order

9. Set up any sorting that you want, then click Next. The Query Wizard - Finish dialog box opens (Figure 12-17).

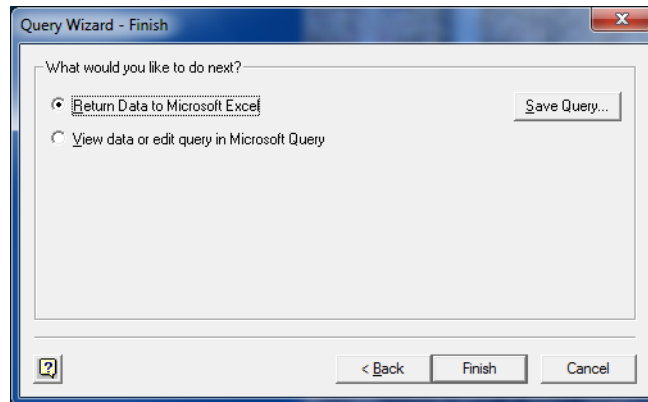


Figure 12-17. Query Wizard - Finish

10. Select the Return Data to Microsoft Excel option and click Finish. The Import Data dialog box opens and a cell in the spreadsheet is highlighted (Figure 12-18).

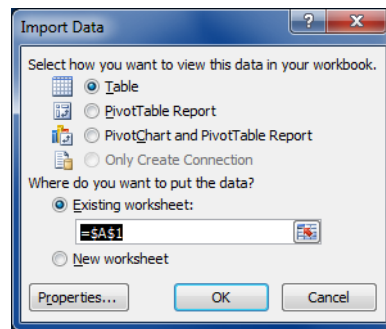
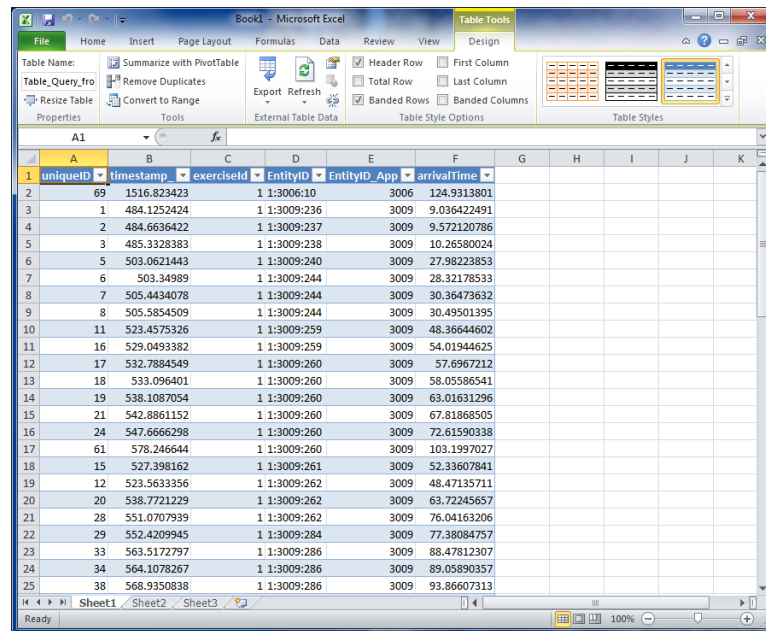


Figure 12-18. Import Data dialog box

11. Specify where you want the data to be imported to and click OK. The data is imported into the spreadsheet (Figure 12-19).



	A	B	C	D	E	F	G	H	I	J	K
1	uniqueID	timestamp	exerciseld	EntityID	EntityID_App	arrivalTime					
2	69	1516.823423		1:1:3006:10	3006	124.9313801					
3	1	484.1252424		1:1:3009:236	3009	9.036422491					
4	2	484.6636422		1:1:3009:237	3009	9.572120786					
5	3	485.3328383		1:1:3009:238	3009	10.26580024					
6	5	503.0621443		1:1:3009:240	3009	27.98223853					
7	6	503.34989		1:1:3009:244	3009	28.32178533					
8	7	505.4434078		1:1:3009:244	3009	30.36473632					
9	8	505.5854509		1:1:3009:244	3009	30.49501395					
10	11	523.4575326		1:1:3009:259	3009	48.36644602					
11	16	529.0493382		1:1:3009:259	3009	54.01944625					
12	17	532.7884549		1:1:3009:260	3009	57.6967212					
13	18	533.096401		1:1:3009:260	3009	58.05586541					
14	19	538.1087054		1:1:3009:260	3009	63.01631296					
15	21	542.8861152		1:1:3009:260	3009	67.81868505					
16	24	547.6666298		1:1:3009:260	3009	72.61590338					
17	61	578.246644		1:1:3009:260	3009	103.1997027					
18	15	527.398162		1:1:3009:261	3009	52.33607841					
19	12	523.5633356		1:1:3009:262	3009	48.47135711					
20	20	538.7721229		1:1:3009:262	3009	63.72245657					
21	28	551.0707939		1:1:3009:262	3009	76.04163206					
22	29	552.4209945		1:1:3009:284	3009	77.38084757					
23	33	563.5172797		1:1:3009:286	3009	88.47812307					
24	34	564.1078267		1:1:3009:286	3009	89.05890357					
25	38	568.9350838		1:1:3009:286	3009	93.86607313					

Figure 12-19. Spreadsheet with data imported

12.5.2. Importing Data from Access to Excel

This section explains how to import data from a Microsoft Access database to Microsoft Excel.

To import data from Access to Excel:

1. Create an MS Access data source as described in [“Configuring a Data Source for Microsoft Access,”](#) on page 12-9.
2. Export a Logger file to Access, as described in [“Exporting Data to a Database,”](#) on page 12-3.
3. Start Excel.
4. Choose **File** → **Open**.
5. Set Files of Type to Access Databases.
6. Select the database that you want to open.
7. Click Open. You may receive a warning prompt. If so, click Open. The Select Table dialog box opens ([Figure 12-20](#)).

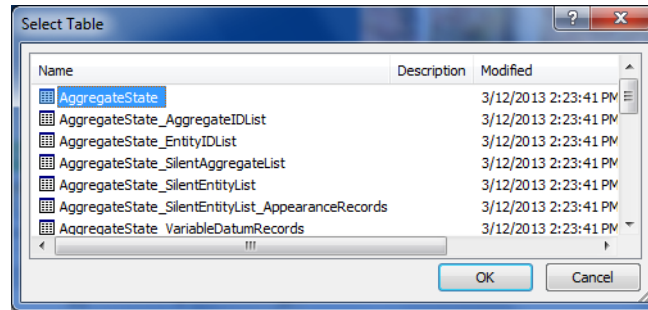


Figure 12-20. Select Table dialog box

8. Select the table that you want to import.
9. Click OK. The data is imported (Figure 12-21).

The image is a screenshot of the Microsoft Excel application window. The title bar says 'Book2 - Microsoft Excel'. The 'Table Tools' ribbon is active, showing 'Table Style Options' and 'Table Styles'. A table named 'Table_MyNewLo' is selected. The table has 10 columns: 'uniqueID', 'arrivalTime', 'timestamp', 'timeStampType', 'exerciseld', 'pduSize', 'EntityID', 'EntityID_Site', 'EntityID_App', and 'EntityID'. The data is organized into 25 rows. The first row is the header, and the subsequent rows contain numerical and text data. The status bar at the bottom shows 'Ready' and 'MyNewLoggerExport'.

Figure 12-21. Logger file data imported from Access to Excel



13. Using Logger Commands

The Logger has a set of commands that cover most Logger operations. They are very versatile in that you can use them on the command line, in the text interface, or in the Logger configuration file. This chapter explains how to use the commands in each of these contexts. It also describes each command.

Using Logger Commands on the Command Line.....	13-2
Using Logger Commands in the Configuration File.....	13-2
Using Logger Commands in the Text Interface.....	13-3
Entering Commands	13-3
Displaying a List of Commands	13-4
Command Descriptions	13-4
Argument Types.....	13-5
System Domain Commands.....	13-6
Player Domain Commands	13-7
Recorder Domain Commands.....	13-9
Simulation Domain Commands.....	13-11
Editor Domain Commands.....	13-19
Command List.....	13-20

13.1. Using Logger Commands on the Command Line

When you start the Logger, you can use the `-r` command-line option to specify Logger commands. Commands must be surrounded by double quotes. To use double quotes within a command, protect them with a backslash escape character (`\`). Strings in string lists are separated by spaces. Strings cannot contain spaces.



Commands that are specified using `-r` are executed after the Logger connects to the exercise. Therefore, commands that need to execute at connection time, such as `SetConnectionParameters`, do not work from the command line.

The following example starts the Logger and opens the file `../logger_tapes/makland2013-DIS.lgr`. Note the use of the backslash to protect the quoted string within the command. (Quotes are not required unless the path has spaces.)

```
loggerversion -r "OpenFile \"../logger_tapes/makland2013-  
DIS.lgr\" 0"
```



The `OpenFile` command is unique. If you simply specify a Logger file on the command line, the Logger assumes `-r OpenFile` and opens the file, for example:

```
loggerversion "../logger_tapes/berkeleyDemoHLA13.lgr"
```

The following example enables time management and specifies a subscription list:

```
loggerversion -n 3 --federateName logger  
-r "SetTimeManagement true true true true 0.1"  
-r "SetSubscriptionLists \"ObjectRoot.BaseEntity.Phys-  
icalEntity.Platform.Aircraft ObjectRoot.BaseEntity.Phys-  
icalEntity.Radio InteractionRoot.WeaponFire\""
```

13.2. Using Logger Commands in the Configuration File

You can put Logger commands in the Logger configuration file. For complete details, please see [“Editing lgrConfig.xml,”](#) on page 5-19.

13.3. Using Logger Commands in the Text Interface

The Logger text interface lets you enter commands in a console window instead of, or in addition to, using the graphical user interface.

- To start the Logger with the console enabled, in the *lgrConfig.xml* file, set the value of the `showConsole` variable to true.

```
<var name="showConsole" type="bool" value="true" />
```

or start the Logger with the `-c` command-line option:

```
loggerversion (-c | --showConsole) 1
```

If you want to, you can disable the GUI. For details, please see [“Running the Logger without a GUI,”](#) on page 6-2.

13.3.1. Entering Commands

The text interface has commands for most Logger functions. Most commands consist of a command and one or more arguments, for example:

```
OpenFile ../logger_tapes/test.lgr 0
```

Commands must include all arguments and the arguments must be typed in the correct order. (For command descriptions, please see [“Command Descriptions,”](#) on page 13-4.) For example, in the `SetTimeManagement` command, you must include values for each time management mode, even if you only want to use one of them:

```
SetTimeManagement true true true true 0.1
```

String list parameters are encoded as quoted strings separated by spaces. Strings in a string list may not have spaces, for example:

```
SetSubscriptionLists "ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft" "ObjectRoot.BaseEntity.PhysicalEntity.Radio" "InteractionRoot.WeaponFire"
```

You can manipulate the command line as follows:

- To execute a command, press **Enter**.
- To reset a command entry, press **Escape**.
- To execute a previously executed command, press the **Up Arrow** key until the command is displayed. Then, press **Enter**.
- If you have displayed a previous command by pressing **Up Arrow**, press **Down Arrow** to move back down the command list.

13.3.2. Displaying a List of Commands

You can display a list of commands. They are displayed in the format:

```
command_name argument_1_name(argument_type) argu-  
ment_2_name(argument_type) ...
```

- To display the list of available commands, type `help` in the console window and press **Enter**.

13.4. Command Descriptions

Commands are organized into domains. If you add a command to *lgrConfig.xml*, you must specify the domain. The domains are:

- ♦ System
- ♦ Player
- ♦ Recorder
- ♦ Editor
- ♦ Simulation
- ♦ Invalid.

When used in the configuration file, simulation domain commands require a protocol string. The valid protocol strings are:

- ♦ None
- ♦ DIS
- ♦ HLA 13
- ♦ HLA 1516
- ♦ HLA 1516e.

13.4.1. Argument Types

Many commands have arguments. The data you enter for arguments must be of the correct type. For example, if an argument takes an int (integer), the value 5 is acceptable, but 5.0 is not. In the command descriptions, command argument types are shown in parentheses following each argument. [Table 13-1](#) describes the argument types used for commands.

Table 13-1: Argument types

Argument Type	Description
int	32 bit signed integer. For example: <code>SetNotifyLevel 2</code>
int64	64 bit signed integer. For example: <code>SetStepSize 1000000</code>
double	64 bit floating point number. For example: <code>SetTimeoutRate 5.0</code>
stringlist	A series of strings, separated by spaces. The individual strings cannot have spaces. The entire series is surrounded by double quotes. For example: <code>"string1 string2 string3"</code>
string	A string of alphanumeric characters.
time	Time values can be entered as a double (in seconds) or in Hour:Minute:Second:Fractions of a second format.
base64	Binary data encoded as a base 64 string.
bool	Boolean value. Can be true, false, 1 (true), or 0 (false).

13.4.2. System Domain Commands

Table 13-2 describes the commands in the System domain.

Table 13-2: System domain commands

Command	Description
DeleteFile <i>filename</i> (string)	Deletes the specified file. This command is considered dangerous and should only be used with caution.
Exit	Shuts down the Logger.
Help	Displays a list of available commands.
LoadPlugin <i>pluginName</i> (string)	Loads the specified plug-in.
SetDefaultTapeDirectory <i>defaultTapeDir</i> (string)	Sets the directory where the Logger creates and loads files.
SetEntityTracking <i>entityTracking</i> (bool)	Allows entity tracking to be processed. Enabled by default. When disabled, entity tracking is not allowed to be processed and the entity filter is empty.
SetFastJumpsWithoutDiff <i>fastJumpsWithoutDiff</i> (bool)	When jumping, checkpoints are used to establish the relative state of objects, but the packets between the checkpoint and jump time are not processed.
SetNotifyLevel <i>vLevel</i> (int) <i>lgrLevel</i> (int)	Sets the notification level of VR-Link and the Logger.
SetPlaybackReadTimeLimit <i>playbackReadTimeLimit</i> (double)	When greater than 0.0, this command sets a limit on how long the Logger can spend reading packets from the Logger file during playback, after which the remaining packets up to the current time will be dropped.
SetTempDir <i>directory</i> (string)	Sets the directory in which the Logger stores temporary Logger files when recording and editing. By default the Logger uses the current user's <i>temp</i> directory (OS specific). If you specify a temporary directory, it is best to use a local directory for performance reasons. To reset the temp directory to the system directory use an empty string. The Logger deletes the temporary files when they are no longer needed.

13.4.3. Player Domain Commands

Table 13-3 describes the commands in the Player domain.

Table 13-3: Player domain commands (Sheet 1 of 2)

Command	Description
<code>ClosePlaybackFile</code>	Closes the current playback file.
<code>OpenFile filename(string) constructFilename (bool)</code>	Opens the specified file for playback or constructs a file name using the specified file name and remote control ID. If the file-name has spaces, you must enclose it in quotes, which need to be protected with backslashes on the command line. For more information, please see File Naming Conventions in the developer guide section of the API documentation.
<code>Pause</code>	Pauses playback. This command has no effect if the Logger does not have a file open or is already paused.
<code>Play</code>	Starts playback. If the Logger is already playing this command has no effect.
<code>PlayPause</code>	Toggles between the paused and play state. If the Logger is stopped, it starts playback.
<code>Restart</code>	Starts playing from the beginning of the file.
<code>RevertFile</code>	Reverts a previously saved, but currently unsaved file to the previously saved state. All changes since the last save are lost. This command applies only to files that you open for playback or editing.
<code>SaveFile reopenAfterSave(bool)</code>	Saves a previously saved, but currently unsaved file to its current file name. The previous file is backed up to a file named <i>file-name.bak.lgr</i> , for example: <i>test.lgr</i> would get saved to <i>test.bak.lgr</i> . The current file contains the latest changes. Saving the file closes it. The <i>reopenAfterSave</i> boolean specifies if the saved file should be reopened after the save.
<code>SaveFileAs filename(string) [description(string) writeDescription(bool) reopenAfterSave(bool)]</code>	Saves the current file to a new file. Saving the file closes it. The <i>reopenAfterSave</i> boolean specifies if the saved file should be reopened after the save. Optionally, add a description.
<code>SetBackupOnSave backup(bool)</code>	Enables or disables automatic backup when you save a previously saved file.
<code>SetCurrentTime time(time)</code>	Moves the timeline cursor to the specified time.

Table 13-3: Player domain commands (Sheet 2 of 2)

Command	Description
<code>SetFollowingInterval interval</code>	Specifies the default length of the zoom timeline, in seconds.
<code>SetFollowingExportInterval interval</code>	Specifies the frequency at which to export timeline data when playing Logger data from an active recording. This also specifies the length, in seconds, of the data recording.
<code>SetLoopAtEnd value(bool)</code>	Enables or disables looping to start at the end of playback.
<code>SetPacketViewInterval interval (double)</code>	Specifies the interval between tick marks in the Packet Count View and Packet Size View.
<code>SetPlaybackSpeed speed(double) skippingPlayback (bool)</code>	Sets the playback speed of the Logger. If <i>skippingPlayback</i> is True, skip from checkpoint to checkpoint without sending the interim state updates.
<code>SetPlayOnLoad value(bool)</code>	Enables starting playback immediately after opening a file.
<code>SetSkippingPlaybackThreshold skippingPlaybackThreshold(double)</code>	Set the playback speed above which skipping playback mode is used automatically.
<code>SetStartStopTimes startTime(time) stopTime(time)</code>	Sets the start and stop time of the Logger. The two times are bound to the first and last times in the file. The stop time must be after the start time.
<code>SkipTime time(double) forward(bool)</code>	Moves the timeline cursor the specified time (forward or backward) relative to the current time.
<code>StopPlayback</code>	Stops playback. This command has no effect if the Logger is not in the playback state.
<code>Unpause</code>	Unpauses the Logger. This command has no effect if the Logger is not paused.

13.4.4. Recorder Domain Commands

Table 13-4 describes the commands in the Recorder domain.

Table 13-4: Recorder domain commands (Sheet 1 of 2)

Command	Description
CancelRecording	Cancels the current recording. The Logger must be currently recording. After the command is executed, the Logger is in an empty state, no file is loaded.
CloseRecordFile	Closes the file currently recording.
NewFile <i>filename(string) description(string) overwrite(bool) constructFilePathAndName(bool)</i>	Opens a new file with the given name and description for recording. The file name is used as given or it is constructed using the specified file name and remote control ID. The description must be enclosed in backslashed quotes if it contains spaces.
PauseRecording	Pauses the recording process. Recording stops, but the data is not saved and the Logger remains in record mode. No data is recorded while the Logger is paused. You can resume recording to the same file.
Record	Starts recording a file that is already loaded. To open a new file and start recording, use <code>newFile</code> .
RecordToMultipleFiles <i>fileName(string)</i>	Starts recording to new files.
SetCheckpointParameters <i>policy(int) size(int) count(int) time(time)</i>	Sets the checkpoint policy parameters on when checkpoints are generated. The Policy is one of the following: • 0 — count, in number of packets. • 1 — size of packets, in bytes. • 2 — time, in seconds between checkpoints. ! You must specify a value for each argument on the command line. The value does not need to be meaningful for the policies that are not being used.
SetPauseOnRecordAs <i>record(bool)</i>	Determines how the Logger behaves when given the Record As command. If true, the Logger pauses and does not begin recording until it is given the Record command. If false, the Logger begins recording as soon as a file name is specified.
SetRecordToMultipleFilesSettings <i>duration(double) namingConvention(string)</i>	Set the multiple file recording parameters.

Table 13-4: Recorder domain commands (Sheet 2 of 2)

Command	Description
<code>SetStopCondition</code> <i>condition</i> (int) <i>close</i> (bool)	Specifies the stop condition for recording as follows: ♦ 0 — no condition. ♦ 1 — duration (enables <code>SetStopDuration</code>). ♦ 2 — filesize (enables <code>SetStopSize</code>). If the <i>close</i> parameter is True, the file is closed when recording is stopped.
<code>SetStopDuration</code> <i>duration</i> (time)	Stops recording at the specified time if <code>SetStopCondition</code> is set to 1.
<code>SetStopSize</code> <i>filesize</i> (int64)	Stops recording when the file reaches <i>filesize</i> size if <code>SetStopCondition</code> is set to 2.
<code>StopRecording</code> <i>closeAndOpenForPlayback</i> (bool)	Stops recording and ends. By default <i>closeAndOpenForPlayback</i> is true (1). If you set <i>closeAndOpenForPlayback</i> to false (0), then after you issue <code>StopRecording</code>, you are still in record mode. You would need to issue the <code>CloseRecordFile</code> command to close the file and exit record mode.
<code>UnpauseRecording</code>	Resume a paused recording.

13.4.5. Simulation Domain Commands

Table 13-5 describes the commands in the Simulation domain. Examples of using the `SetConnectionParameters` command are in the protocol-specific configuration files. For a full definition of the arguments, please see Chapter 5, [Configuring the Logger](#).

Table 13-5: Simulation domain commands (Sheet 1 of 8)

Command	Description
Protocol-independent Commands	
<code>Connect</code>	Connects to the network simulation using the previously set parameters.
<code>Disconnect</code>	Disconnects from the network simulation.
<code>FilterEntityTypes <i>entityType1</i> ... <i>entityTypeN</i>(string)</code>	Specifies a list of entity types to list in the Filter by Entity Type dialog box and enables filtering for these entity types.
<code>SetBackChannelConnection <i>useBackChannel</i>(bool) <i>address</i>(string) <i>port</i>(int) <i>interface</i>(string)</code>	Set the parameters for the back channel connection configuration.
<code>SetCommonEntityTypes <i>entityType1</i> ... <i>entityTypeN</i>(string)</code>	Specifies a list of entity types to list in the Filter by Entity Type dialog box. Does not enable filtering of these entity types.
<code>SetListenWhenIdle <i>state</i> (bool)</code>	Enables or disables listening to the network when the Logger is neither recording or playing a file.
<code>SetRemoteControlId <i>remoteControlId</i> (integer)</code>	Specifies the Logger's remote control ID. By default the value is assigned the Logger's site ID. This value is used if <code>SetRemoteControlSettings allowAllRemoteControl</code> is disabled.

Table 13-5: Simulation domain commands (Sheet 2 of 8)

Command	Description
SetRemoteControlSettings <i>process</i> (bool) <i>record</i> (bool) <i>allowDelete</i> (bool) <i>allowAll</i> (bool) <i>sendResponses</i> (bool) <i>crippleGui</i> (bool) <i>useLegacyPdu</i> (bool) <i>connectSendInterval</i> (double) <i>heartbeatSendInterval</i> (double)	Specifies how the Logger responds to Logger Control messages. ♦ <i>process</i> determines whether the Logger responds to Logger control messages. ♦ <i>record</i> determines whether or not Logger control messages are included in the file. ♦ <i>allowDelete</i> determines whether or not to respond to the delete file Logger control message. ♦ <i>allowAll</i> determines whether remote control PDUs are filtered based on the Logger's site-application-entNum triplet, where the Logger's remote control ID is used for the entNum. By default this parameter is enabled and all remote control PDUs are accepted. When disabled, the Logger accepts remote control PDUs when the receiver ID matches the Logger. The receiver ID can have wildcard values (0xFFFF or unsigned int = 65535) in any of the triplet values (for example, 1:1:65535 matches any Logger with site = 1 and application ID = 1). Remote control PDUs are filtered when they do not match. By default, the Logger's remote control ID equals its site ID. The remote control ID can be changed using the SetRemoteControlId command. ♦ <i>sendResponses</i> specifies whether the Logger sends the internal responses over the network for the remote control interface. The responses are generated by the Logger to provide feedback on the changes in state of the Logger based on the execution of commands (for example, fileOpened, playback paused, and so on). ♦ <i>crippleGui</i> disables some features in the GUI and causes it to start up minimized. It forces the data views to not be visible at startup and it disables mouse clicks in the timeline (that is, you cannot jump via mouse clicks). It also disables pop up error messages. ♦ <i>useLegacyPdu</i> If set to 0, use Logger Control Toolkit PDUs and interactions from the Logger Control API introduced in Logger 5.1. If set to 1, use PDUs and interactions from the Logger Control Toolkit API in Logger 5.0 and earlier. The Logger can only use one API at a time. ♦ <i>connectSendInterval</i> specifies how frequently, in seconds, to send connect messages. The connect message is used by a remote control interface to discover Loggers. Once the Logger is connected, the Logger stops sending the connect message and start sending a heartbeat message. If the connectSendInterval is less equal 0, then the connect message is not sent.

Table 13-5: Simulation domain commands (Sheet 3 of 8)

Command	Description
SetRemoteControlSettings (continued)	♦ <i>heartbeatSendInterval</i> specifies how frequently, in seconds, to send a heartbeat message after the Logger has connected. The heartbeat message is used by a remote control interface to determine if the Logger is no longer responsive (for example, crashed or broken network connection). If the <i>heartbeatSendInterval</i> is less than 0, then the heartbeat message is not sent.
SetSaveConnectionConfiguration	<i>saveConnectionConfiguration</i> (bool)
	Enables or disables saving of connection information to the <i>lgrConfig.xml</i> file.
SetScaleAttributes	<i>scale</i> (bool)
	Enables or disables attribute scaling due to changes in playback speed. When enabled, attributes (such as velocity and acceleration) are scaled to synchronize with playback speed changes. When disabled no scaling occurs.
SetSendOnlyUpdatedValues	<i>state</i> (bool)
	If true, when the Logger receives attribute value update requests, only send updates for values that have been updated in the Logger file. Do not send default values.
SetVRFFiltering	<i>filter</i> (bool) <i>loId</i> (int) <i>hiId</i> (int)
	Configures VR-Forces message filtering. The filter argument enables or disables filtering of VR-Forces messages. The <i>loId</i> argument specifies the lowest message ID to filter out. The <i>hiId</i> argument specifies the highest message ID to filter out.
HLA-specific Commands	
CommonObjectNames	<i>objectName1</i> ... <i>objectNameN</i> (string)
	Specifies a list of object names to list in the Filter by Object Names dialog box. Does not enable filtering of these object names.
EnableRequestUpdates	<i>state</i> (bool)
	Enable or disable whether request updates is invoked when an object is discovered. Default: true.
FilterObjectNames	<i>objectName1</i> ... <i>objectNameN</i> (string)
	Specifies a list of object names to list in the Filter by Object Names dialog box and enables filtering for these entity object names.
LoadSubscriptionListFile	<i>filename</i> (string)
	Loads a subscription list file. The subscription list file is an XML file that you create using the Logger GUI. The Logger must be connected. The subscription settings are loaded from the file.

Table 13-5: Simulation domain commands (Sheet 4 of 8)

Command	Description
<code>SaveSubscriptionListFile filename(string)</code>	Saves the subscription settings to an XML file with the specified name. In the GUI, the extension <i>.subx</i> is added to filenames, but it is not added to the filename when saved from the command line.
<code>SetConnectionParametersHLA federateType(string) federationName(string) fedFile(string) ignoreAdvisories(bool) fomVersion(double) fomRevision(int) fomMapper(string) fomMapperData(base64) useEmptyFomMapper(bool) useMatrexFomMapper(bool) matrexFomMapperData(string)</code>	Specifies HLA (1.3 and 1516) or HLA Evolved connection parameters:
	♦ <i>federateName</i>. The name of the Logger federate. (HLA Evolved only) ♦ <i>federateType</i>. The type of the Logger federate. ♦ <i>federationName</i>. The name of the federation execution. ♦ <i>fedFile</i>. The name of the FED file (or FDD or XML file). ♦ <i>fomModules</i>. One or more FOM module files. (HLA Evolved only) ♦ <i>MIM</i>. A MIM file. (HLA Evolved only) ♦ <i>ignoreAdvisories</i>. Configures the Logger to ignore RTI advisory queues and always send data. Default: false. ♦ <i>fomVersion</i>. The RPR FOM version being used. ♦ <i>fomRevision</i>. The RPR FOM revision being used. ♦ <i>fomMapper</i>. The name of the FOM Mapper being used, if any. ♦ <i>fomMapperData</i>. Initialization data for the FOM Mapper, if required. ♦ <i>useMatrexFomMapper</i>. Use the Matrex FOM Mapper. ♦ <i>matrexFomMapperData</i>. Initialization data for the Matrex FOM Mapper. ♦ <i>useEmptyFomMapper</i>. The Logger will use an empty FOM Mapper, allowing it to record and play back using any FOM. The empty FOM Mapper does not support some playback features such as pause or scaling, which require the Logger to use dead reckoning to generate intermediate updates.

Table 13-5: Simulation domain commands (Sheet 5 of 8)

Command	Description
<code>SetDdmParameters</code>	<code>enableDdm(bool)</code> <code>playbackMode(string)</code> <code>subscribeMode(string)</code> <code>playbackDynamicAssociation(bool)</code> <code>playbackDynamicExtents(bool)</code> <code>allowPlaybackWithoutRegions(bool)</code> <code>recordConveyedRegions(bool)</code> <code>requestUpdates(bool)</code> <code>subscribeUnspecifiedAttributes(bool)</code> Specifies DDM parameters, as follows: ♦ <code>enableDdm</code>. Enable record and playback using DDM. Default: false. ♦ <code>playbackMode</code>. Supported modes are: – <code>PlaybackRegionsUseConveyedExtents</code>. During playback, regions are created and updated using recorded region data. – <code>PlaybackRegionsUseData</code>. Regions are created and updated by inspecting simulation data (. Requires plug-in extension. – <code>PlaybackRegionNotEnabled</code> (default). Updates and interactions are sent without regions. ♦ <code>subscribeMode</code>. Supported modes: – <code>SubscribeRegionsAreGlobal</code>. DDM subscriptions are made with global regions with extents of global min and max. – <code>SubscribeRegionsAreSpecified</code>. The regions are specified through subscription configuration files. (For details, please see the <code>LoadSubscriptionListFile</code> command.) The subscription region configuration can also be specified programmatically through the Logger plug-in API. – <code>SubscribeRegionNotEnabled</code> (default). Subscriptions are performed without regions. ♦ <code>playbackDynamicAssociation</code>. During playback, the association of regions with object instance attributes can be fixed or dynamic. If it is known that region associations with attribute instances never change, the overhead of checking for changes can be eliminated. This parameter should be disabled if possible. Default: false. ♦ <code>playbackDynamicExtents</code>. During playback, the region extents can be fixed or dynamic. The conventional notion of DDM regions regards them as dynamically changing extents to reflect the state of simulation data (for example, surrounding an entity's position). An alternative approach is for the regions to be fixed and the object attribute instances move from region to region. If it is known that region extents never change, the overhead of checking for changes can be eliminated. This parameter should be disabled if possible. Default: false. ♦ <code>allowPlaybackWithoutRegions</code>. During playback, it may happen that a region association cannot be established. In these cases, the Logger sends the data without region association. Disable this parameter if the federation agreement requires that all data must be transmitted with associated regions. Default: true.

Table 13-5: Simulation domain commands (Sheet 6 of 8)

Command	Description
	♦ <i>recordConveyedRegions</i>. Enable this parameter to record conveyed regions. Default: true. ♦ <i>requestUpdates</i>. Requesting updates can result in spikes in network traffic, especially since it is often performed by late joining federates when the exercise is already populated with many objects. In some DDM federations, object heartbeats are used instead of update requests. Disable this parameter to prevent the Logger from making requests for updates. ♦ <i>subscribeUnspecifiedAttributes</i>. The Logger may not be able to determine a region association to apply to an interaction class or object class attribute subscription. This situation may occur if the interaction or class does not specify a routing space or dimension in the FOM. It can also happen if the <i>subscribeMode</i> is configured as <i>SubscribeRegionsAreSpecified</i>, but some classes or attributes are left unspecified. By default, the Logger subscribes to these interaction classes and object class attributes without using a region association (in effect a global subscription). Disable this parameter if the Logger should only subscribe when a region association can be used. Default: true.
<code>SetDisableTimeManagementTest <i>disableTesting</i>(bool)</code>	Disables testing for time management if that test causes the Logger to behave badly. Ordinarily, when time management is enabled, when the Logger connects, it tries to become a time constrained federate to see if time management is enabled in the RTI. If it succeeds, the Logger sets time management settings as configured. In some cases, this test causes the Logger to hang. If this problem occurs, setting <i>disableTesting</i> to true turns off the test. However, time management is disabled.
<code>SetDrainInput <i>maxTime</i>(double), <i>minTime</i>(double)</code>	Specifies the minimum and maximum amount of time, in seconds, to spend reading messages from the exercise connection (that is, the network) each tick.
<code>SetDynamicNamePrefix <i>prefix</i>(string)</code>	Specifies the prefix that the Logger attaches to the beginning of object names that begin with HLA_ when it is recording exercises.
<code>SetHeartbeatIntervals "className(string):interval(float):inheritable(bool) ..."</code>	Specifies a heartbeat interval for playback. You can specify multiple classes. For details, please see “Configuring HLA Heartbeating,” on page 5-8.
<code>SetIgnoreAdvisories <i>mode</i>(bool)</code>	Specifies the Logger’s response to HLA advisory messages. You must specify this command before the Connect command. Default: false.

Table 13-5: Simulation domain commands (Sheet 7 of 8)

Command	Description
<code>SetPassiveSubscription mode(bool)</code>	Configures passive subscription for HLA. You must specify this command before the Connect command. Default: false.
<code>SetSubscriptionLists objectClasses(stringList) interactionClasses(stringList)</code>	Sets the class subscription list. To subscribe to a class, the class must be in the list. Each class must be specified using the full HLA class name (including parents). Some operating systems will not allow command line buffers that are long enough to allow use of this command on the command line. The Logger must be connected. The updated subscription list takes effect the next time you record.
<code>SetTimeManagement timeConstrainedRecording(bool) timeConstrainedPlayback(bool) timeRegulatingRecording(bool) timeRegulatingPlayback(bool) lookahead(double)</code>	Configures time management options. The <i>timeConstrainedPlayback</i> and <i>timeRegulatingRecording</i> arguments must be included on the command line, but are not supported and are ignored.
<code>SetTimeoutIntervals "className(string):interval(float):inheritable(bool) ..."</code>	Specifies an object timeout for recording. You can specify multiple classes. For details, please see “Configuring HLA Timeouts,” on page 5-9.
DIS-specific Commands	
<code>AddCustomPDU pduName(string) pduKind(string) entityIdOffset(string) entityTypeOffset(string)</code>	Specifies a PDU to add to the Filter by PDU Kind dialog box.
<code>CommonEntityIds id1 ... idN(string)</code>	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box. Does not enable filtering of these entity IDs.
<code>FilterEntityIds id1 ... idN(string)</code>	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box and enables filtering for these entity IDs.

Table 13-5: Simulation domain commands (Sheet 8 of 8)

Command	Description
SetConnectionParameters <i>port</i> (int) <i>exerciseId</i> (int) <i>siteId</i> (int) <i>appNumber</i> (int) <i>useAsynchIO</i> (bool) <i>sendBufferSize</i> (int) <i>recvBufferSize</i> (int) <i>multicastTtl</i> (int) <i>destAddr</i> (string) <i>subnetMask</i> (string) <i>deviceAddr</i> (string) <i>mcastAddresses</i> (stringList)	Specifies parameters for the DIS exercise connection, as follows: ♦ <i>port</i>. The port to listen to. ♦ <i>exerciseId</i>. The Logger's exercise ID. ♦ <i>siteId</i>. The Logger's site ID. ♦ <i>appNumber</i>. The Logger's application number. ♦ <i>useAsynchIO</i>. Enable (1) or disable (0) asynchronous I/O. ♦ <i>sendBufferSize</i>. The size of the send buffer. ♦ <i>recvBufferSize</i>. The size of the receive buffer. ♦ <i>multicastTtl</i>. The multicast time-to-live, in seconds. ♦ <i>destAddr</i>. The destination address to send to. ♦ <i>subnetMask</i>. The subnet mask used for determining the destination when <i>destAddr</i> is not unspecified. ♦ <i>deviceAddr</i>. The network interface address to use. ♦ <i>mcastAddresses</i>. A list of multicast addresses.
SetDrainInput <i>maxTime</i> (double), <i>numPdusToRead</i> (int)	Specifies the maximum amount of time, in seconds, to spend reading messages from the exercise connection (that is, the network) and the number of PDUs to read between each check of the maximum time.
SetHeartbeatRate <i>rate</i> (double)	Specifies the DIS heartbeat rate.
SetTimeoutRate <i>rate</i> (double)	Specifies the DIS timeout rate.

13.4.6. Editor Domain Commands

Table 13-6 describes the commands in the Editor domain.



Editing operations can be performed only on files that have been opened for playback. After you perform an edit, the edited file is unsaved. You must save it to preserve the change.

Table 13-6: Editor domain commands (Sheet 1 of 2)

Command	Description
<code>AppendFileToEnd filename(string)</code>	Appends the file <i>filename</i> to the end of the open file.
<code>ClearSection start(time) end(time)</code>	Removes all packets between the <i>start</i> and <i>end</i> times in a file.
<code>ConvertFile</code>	Converts the current file to the latest file format. This command is not valid for files that are already in the latest file format.
<code>CutSection start(time) end(time)</code>	Removes all packets between the <i>start</i> and <i>end</i> times in a file. All packets after the <i>end</i> time are offset so that the first packet after the <i>end</i> time is now at the <i>start</i> time.
<code>ExportFile exportFormat(string) destination(string) checkpoint(bool) openAfterExport(bool)</code>	Exports a file using a given <i>exportFormat</i> to a <i>destination</i>. The last two parameters are only used when exporting a Logger file. The Logger tries to export using the <i>exportFormat</i> and the <i>destination</i>. If <i>checkpoint</i> is true and the file is a Logger file, new checkpoints are generated for the output. If <i>openAfterExport</i> is true and the file is a Logger file, the Logger closes the current file and tries to open the new file. The valid <i>exportFormat</i> strings are based upon the registered export objects. The valid strings are identical to the Export and Export Selection menu commands in the GUI. The formats supplied with the Logger are Plain Text (if the lgrDump plug-in is loaded) and Logger File (export to a Logger file). The <i>destination</i> typically is a filename (but does not need to be) and is specific to the <i>exportFormat</i>. The Plain Text exporter exports to any filename, but is typically exported to a <i>.txt</i> file. The Logger File exporter exports to a <i>.lgr</i> file and adds that extension if it is not in the <i>destination</i> string. Other exporters may encode the <i>destination</i> differently.

Table 13-6: Editor domain commands (Sheet 2 of 2)

Command	Description
<code>ExportSection exportFormat(string) destination(string) start(time) end(time) offset(double) checkpoint(bool) openAfterExport(bool)</code>	Exports a section of a file using a given <i>exportFormat</i> to a <i>destination</i> . This command works just like the <code>ExportFile</code> command except that it exports a segment of a file rather than the entire file.
<code>MergeInFile filename(string) offset(double)</code>	Merges the Logger file <i>filename</i> into the open file. An optional offset is applied to all packets from the file before being combined with the current data.
<code>OffsetTime offset(double)</code>	Offsets the time of the packets in the open file by the <i>offset</i> value. The <i>offset</i> value can be positive or negative. However, if you specify a negative offset, any data that results with a time less than zero is lost.
<code>PrefixFileToStart filename(string)</code>	Inserts the file named <i>filename</i> at the start of the open file.
<code>TrimStart</code>	Removes any dead time from the start of the file and offsets all packets so that the first packet time stamp is 0.0.

13.5. Command List

Table 13-7 section lists all Logger commands and their domain:

Table 13-7: Logger commands

Command	Domain
<code>AddCustomPDU</code>	Simulation
<code>AppendFileToEnd</code>	Editor
<code>CancelRecording</code>	Recorder
<code>ClearSection</code>	Editor
<code>ClosePlaybackFile</code>	Player
<code>CloseRecordFile</code>	Recorder
<code>CommonEntityIds</code>	Simulation
<code>CommonObjectNames</code>	Simulation
<code>Connect</code>	Simulation
<code>ConvertFile</code>	Editor
<code>DeleteFile</code>	System

Table 13-7: Logger commands

Command	Domain
Disconnect	Simulation
EnableRequestUpdates	Simulation
ExportFile	Editor
ExportSection	Editor
Exit	System
FilterEntityTypes	Simulation
FilterObjectNames	Simulation
Help	System
LoadPlugin	System
LoadSubscriptionListFile	Simulation
MergeInFile	Editor
NewFile	Recorder
OffsetTime	Editor
OpenFile	Player
Pause	Player
PauseRecording	Recorder
Play	Player
PlayPause	Player
PrefixFileToStart	Editor
Record	Recorder
RecordToMultipleFiles	Recorder
Restart	Player
RevertFile	Player
SaveFile	Player
SaveFileAs	Player
SetBackChannelConnection	Simulation
SaveSubscriptionListFile	Simulation
SetBackupOnSave	Player
SetCheckpointParameters	Recorder
SetCommonEntityTypes	Simulation
SetConnectionParametersHLA	Simulation
SetConnectionParametersHLAe	Simulation
SetCurrentTime	Player

Table 13-7: Logger commands

Command	Domain
SetDdmParameters	Simulation
SetDefaultTapeDirectory	System
SetDisableTimeManagementTest	Simulation
SetDrainInput	Simulation
SetDynamicNamePrefix	Simulation
SetEntityTracking	System
SetFastJumpsWithoutDiff	System
SetFollowingInterval	Player
SetFollowingExportInterval	Player
SetHeartbeatRate	Simulation
SetIgnoreAdvisories	Simulation
SetListenWhenIdle	Simulation
SetLoopAtEnd	Player
SetNotifyLevel	System
SetPacketViewInterval	Player
SetPassiveSubscription	Simulation
SetPauseOnRecordAs	Recorder
SetPlaybackReadTimeLimit	System
SetPlaybackSpeed	Player
SetPlayOnLoad	Player
SetRecordToMultipleFilesSettings	Recorder
SetRemoteControlId	Simulation
SetRemoteControlSettings	Simulation
SetSaveConnectionConfiguration	Simulation
SetScaleAttributes	Simulation
SetSendOnlyUpdatedValues	Simulation
SetSkippingPlaybackThreshold	Player
SetStartStopTimes	Player
SetStopCondition	Recorder
SetStopDuration	Recorder
SetStopSize	Recorder
SetSubscriptionLists	Simulation
SetTempDir	System

Table 13-7: Logger commands

Command	Domain
SetTimeoutRate	Simulation
SetVRFFiltering	Simulation
SkipTime	Player
StopPlayback	Player
StopRecording	Recorder
TrimStart	Editor
Unpause	Player
UnpauseRecording	Recorder



14. Using the Logger Remote Control

This chapter explains the purpose of the Logger Remote Control, how to configure it, and how to use it to record and play back simulation data.

The Logger Remote Control	14-2
Configuring Loggers to be Controlled Remotely.....	14-2
Starting the Logger Remote Control	14-3
Recording Exercises with the Logger Remote Control.....	14-3
Playing Exercises with the Logger Remote Control	14-5
Configuring the Logger Remote Control Connection.....	14-6
Specifying the Refresh Rate.....	14-6
Setting the Timebar Color	14-7
Exiting the Logger Remote Control and Connected Loggers	14-7
Logger Remote Control Command-line Options.....	14-8

14.1. The Logger Remote Control

The Logger Remote Control lets you control one or more Loggers from one location. You can record data and play it back from the Logger Remote Control. If you are running more than one remote Logger, you cannot control them independently of each other. The commands affect all of them at the same time.

The primary use for the Logger Remote Control is to provide one point of control when you are using multiple Loggers to record a large exercise. To improve performance and keep Logger files at reasonable sizes, Logger users often use multiple Loggers, each of which uses the Logger's filtering feature to record a portion of the network traffic. The main problem with this approach is getting the various Loggers to start and stop recording and playback at the same time. The Logger Remote Control solves this problem.

The Logger Remote Control is a separate program – *./bin/lgrControlGui*. It is protocol-independent. It can control DIS Loggers, HLA Loggers, or both.

14.2. Configuring Loggers to be Controlled Remotely

By default, Loggers are not configured to be controlled using the Logger Remote Control. You must make some changes to the configuration file.

To enable remote control set the following variables to true, as shown:

```
<var name="processRemoteControl" type="bool" value="true"/>
<var name="useBackChannel" type="bool" value="true"/>
```

Each Logger must have a unique remote control ID, which is specified as follows:

```
<var name="remoteControlId" type="int" value="2"/>
```

The Logger Remote Control controls Loggers whose back channel port matches its connection port. The default back channel port is 4053. To change the port, edit the **backChannelPort** variable in the Logger configuration file:

```
<var name="backChannelPort" type="int" value="4053"/>
```



A Logger's DIS port (**disPort** variable) is completely separate from the back channel port. You can have multiple Logger's using different DIS ports and running in different DIS exercises and if remote control is enabled, they will all be controlled by the same Logger Remote Control unless you specify a different port for the Logger Remote Control and a matching back channel port for the Loggers it is supposed to control.

14.3. Starting the Logger Remote Control

- To start the Logger Remote Control on Windows, on the Start Menu, choose **All Programs** → **MÄK Technologies** → **Logger 5.4** → **Logger Remote Control**. The Logger Remote Control opens (Figure 14-1).

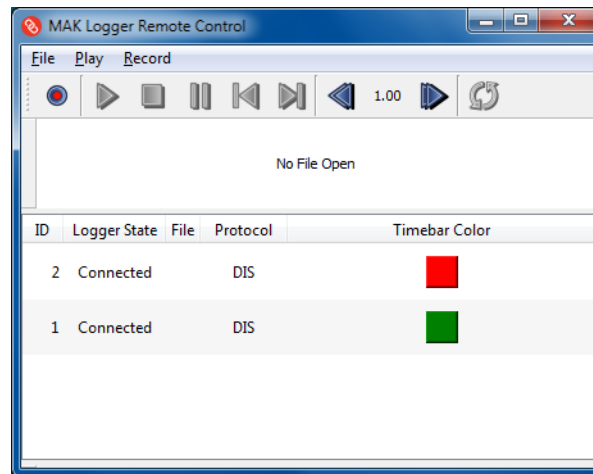


Figure 14-1. Logger Remote Control

To start the remote Logger GUI on Linux or from the Windows command line:

1. Change directory to `./bin64`.
2. Run `lgrControlGui`.

14.4. Recording Exercises with the Logger Remote Control

When you record an exercise from the Logger Remote Control, each Logger records a standard `.lgr` file using a name based on the name you specified in the Record As dialog box (the base-name). The naming convention and location of the file is determined as follows:

- ♦ If the Logger Remote Control is connected to multiple Loggers, it creates a subdirectory of the default Logger recording location using the base-name. The new Logger recording is saved in this directory. It is named `base-name_remoteControlID`. For example, if the base-name is `myTest`, the Logger with a remote control ID of 1 records the file to `./loggertapes/myTest/myTest_1.lgr` and the Logger with a remote control ID of 3 records the file to `./loggertapes/myTest/myTest_3.lgr`.
- ♦ If the Logger Remote Control is connected to one Logger, you can save the recording using the same convention as for multiple Loggers by selecting the Construct File Name check box.
- ♦ If the Logger Remote Control is connected to one Logger and you do not select the Construct File Name check box, the file is saved to the default location or any location you select using the Browser button. It is saved as `base-name.lgr`.

The Logger Remote Control also records a file in its local *logger_tapes* directory. This file has the extension *.rclgr*. This file does not record any simulation data. It records the details of the Loggers that were controlled by the Logger Remote Control when the recording was made. It uses this information to control the correct Loggers when you play back the recording.

To record an exercise using the Logger Remote Control:

1. Choose **Record** → **Record As**, or click the Record button (●) on the toolbar. The Recording File Name dialog box opens.

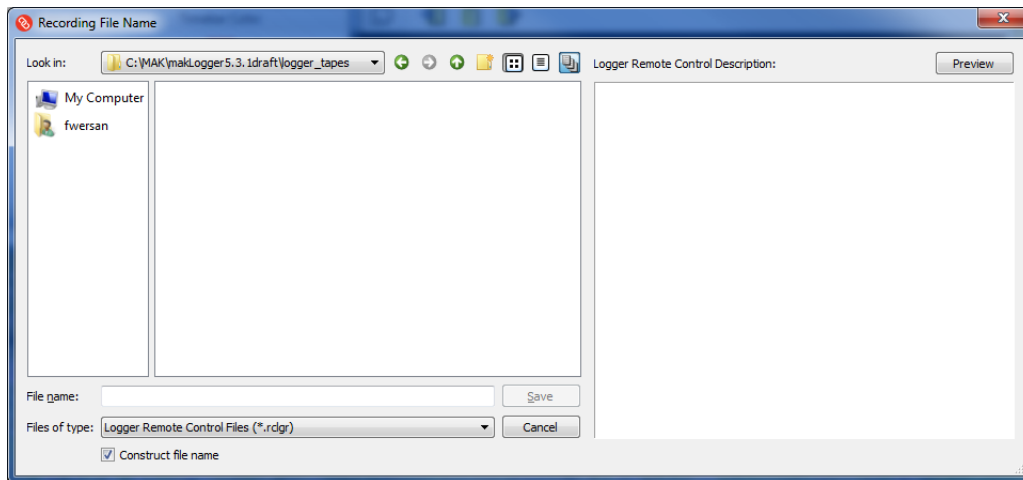


Figure 14-2. Recording File Name dialog box

2. Type a name for the recording (the base-name).

If you have multiple Loggers connected, the Logger Remote Control creates a new directory for each Logger using the base-name and creates a file in that directory in the format *base-name_remoteControlID.lgr*. (The Construct File Name check box is unavailable because this option is mandatory for multiple Loggers.)

If only one Logger is connected and you just enter a base-name, the file is saved in the default recording directory as *base-name.lgr*. (You can also click Browse and choose the directory in which to save it, or type a complete pathname, or you can select Construct File Name and the Logger Remote Control will create a directory and file name using the same format as it does for multiple Loggers.)

3. Optionally, add a description, as follows:
 - a. In the upper right corner of the dialog box, click Edit. The description window becomes editable.
 - b. Type the description. You can type simple text or use HTML coding.
 - c. To see the resulting description, click Preview.
4. Click Save.

5. In the Logger Remote Control window, click the Record button.
6. Control the recording process as you would in a Logger window.

14.5. Playing Exercises with the Logger Remote Control

The process for playing files through the Logger Remote Control is similar to that for playing files directly in the Logger. The difference is that (except as noted below) rather than selecting a recording with a *.lgr* extension, you select a *.rclgr* file. When you load the *.rclgr* file in the Logger Remote Control, it sends a message to each Logger and instructs it to load its local version of the *.lgr* file.



If the Logger Remote Control is controlling just one Logger, you have the option of opening a regular Logger file (*.lgr*) located in the Logger's default directory, rather than a local Logger Remote Control file (*.rclgr*).

To play a Logger file using the Logger Remote Control:

1. Choose **File** → **Open**. The Open File dialog box opens.
2. Type the name of the file you want to open, or click Browse and select a file in the Open File dialog box.
 - If the Logger Remote Control is connected to multiple Loggers, the file must have the extension *.rclgr*.
 - If the Logger Remote Control is connected to one Logger and you want to load a Logger file that does not have a *.rclgr* file associated with it, select Bypass Remote Control File. (This option is disabled if the Logger Remote Control is connected to multiple Loggers.)
3. Click OK.
4. Choose **Play** → **Play/Pause**, or click the Play button (▶) on the toolbar.
5. Manage playback using the Play controls as you would in a Logger window.

14.6. Configuring the Logger Remote Control Connection

A Logger Remote Control connection has parameters that you can change. The default port is 4053. If you want to change the port, you have to change it in the Logger Remote Control and in the configuration file for each remotely controlled Logger.

The Destination Address must match the value of the backChannelAddress variable in the configuration files for the remote Loggers (default: 229.7.7.23).

To edit a Logger Remote Control's connection:

1. In the Logger Remote Control, choose **File** → **Connection**. The Connection dialog box opens (Figure 14-3).

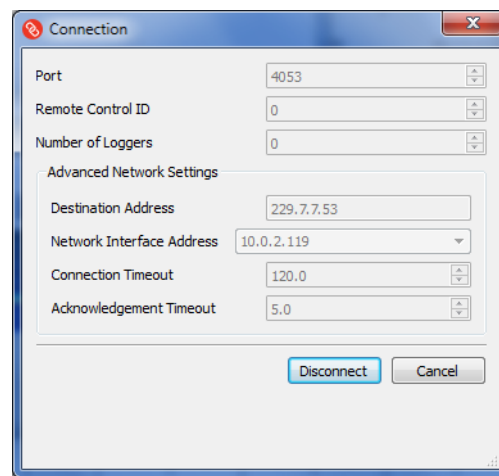


Figure 14-3. Connection dialog box

2. If the Logger Remote Control is connected, none of the parameters are editable. Click Disconnect.
3. Change any of the parameters that you want to change.
4. Click Connect.

14.7. Specifying the Refresh Rate

The Logger Remote Control refresh rate determines how frequently the Logger Remote Control gets the current time from the remote Loggers. Decreasing the frequency reduces network traffic, but means that the timeline in the Logger Remote Control advances less smoothly. The default is three times per second.

To specify the refresh rate:

1. Choose **File** → **Refresh Rate**. The Refresh Rate dialog box opens.
2. Enter a value, in updates per second.

3. Click OK.

14.8. Setting the Timebar Color

Each connected Logger is given a timebar color. When you record or play a file, a timebar for each Logger is displayed in the Logger Remote Control timeline. If the Loggers are recording or playing at the same rate, you will not see any differentiation from the different time bars, but if a Logger stops recording or is delayed for some reason, you will see the individual timebars (Figure 14-4). This will alert you to the possibility that there is a problem with one of the Loggers.

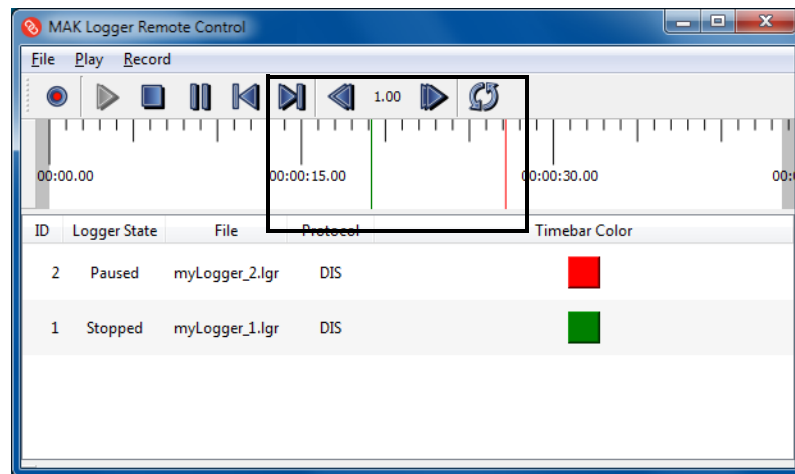


Figure 14-4. Logger Remote Control timebars

To change the color of the timebar for a Logger:

1. Click the color swatch that you want to change. A color picker dialog box opens.
2. Select the new color.
3. Click OK.

14.9. Exiting the Logger Remote Control and Connected Loggers

When you shut down the Logger Remote Control you can also shut down the connected Loggers at the same time.

- To shut down the Logger Remote Control and connected Loggers, choose **File** → **Exit and Quit Loggers**.

14.10. Logger Remote Control Command-line Options

You can run the Logger Remote Control from the command line. [Table 14-1](#) lists command-line options.

Table 14-1: Logger Remote Control command-line options

Option	Description
(-- --ignore_rest)	Ignores the rest of the labeled arguments following this flag.
--acknowledgeTimeout <i>timeout</i>	The amount of time, in seconds, that the Logger Remote Control waits for an acknowledgment message from the Loggers to which it sends commands. If an acknowledgement is not received within this timeout interval, the command is considered lost.
--backChannelAddress <i>address</i>	The destination address for back channel communication.
--backChannelInterface <i>address</i>	The interface to use for the back channel communication.
--backChannelPort <i>port</i>	The port used by the Logger Remote Control. It must be the same for the Logger Remote Control and all connected Loggers. Default: 4053.
--connectionTimeout <i>timeout</i>	The amount of time, in seconds, to wait before timing out a connection. Default: 120 seconds.
(-h --help)	Displays usage information and exits.
--numberOfLoggers <i>number</i>	The maximum number of Loggers to connect to. 0 means connect to all available Loggers.
--remoteControlId <i>ID</i>	The ID of the Logger Remote Control. Must be unique among all running Loggers.
(-v --version)	Displays version information and exits.



A. Troubleshooting

This appendix describes how to identify and correct the most common network problems that prevent the Logger from sending and receiving data properly.

Troubleshooting Logger Network Communications (DIS Only)	A-2
Using the Correct Broadcast Address and UDP Port	A-2

A.1. Troubleshooting Logger Network Communications (DIS Only)

In a typical DIS exercise, VR-Link applications (including the Logger) communicate with other applications using broadcast UDP/IP over a local area network. All applications in the exercise use the same IP broadcast address, and each application's host machine has a network interface device that is configured for that address. All machines must share a common netmask.

In addition, all participating applications must use the same UDP port.

If the Logger uses an IP address, netmask or port that is different from that used by your other applications, communication problems occur.

A.1.1. Using the Correct Broadcast Address and UDP Port

By default, the Logger communicates to the host machine's default broadcast address, using UDP port 3000. If the Logger appears to start normally, but fails to communicate with other DIS applications, the Logger may be using an incorrect broadcast address or UDP port. You can correct this problem by running the Logger with arguments that explicitly specify the correct address and port:

1. Determine the broadcast address used by your other DIS applications. This may be the network's default broadcast address, or may be specified through the application's startup command or configuration file at startup. For more information, please see your System Administrator, or use the Linux `ifconfig` command to display the broadcast address.
2. Determine the UDP port used by your other applications.
3. Start the Logger and notice the status message that appears at startup:

```
NetSocket ok for port 3000 to 123.456.789.255
```

4. If the port and address in the status message do not match those used by your other applications, exit the Logger and restart using the Address option (`-A`), the DIS port option (`-P`), or both, to specify the appropriate values. Enter:

```
loggerversion -A address -P port
```

where address and portnum are the broadcast address and UDP port used by your other applications, for example:

```
loggerversion -A 123.456.999.255 -P 3020
```



MÄK Products Glossary

This glossary defines terms used in MÄK product documentation.

absolute dead reckoning

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

aggregate

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

API

Application Programming Interface.

articulated part

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

attached part

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

body coordinate frame

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

channel

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

clipping

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

clipping planes

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

culling

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

Cartesian coordinates

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

Data Distribution Management

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

dead-reckoning

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

DDM

[Data Distribution Management.](#)

Defense Modeling and Simulation Office

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

DIS

[Distributed Interactive Simulation.](#)

DIS data packets

See [Protocol Data Unit](#).

display engine

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

Distributed Interactive Simulation

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

The DIS protocol has been superseded by the [High-Level Architecture](#) for use by the Department of Defense.

DMSO

See [Defense Modeling and Simulation Office](#).

element definition

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

emitter

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

entity

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

entity maintenance

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

entity-type

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

The components of an entity-type are:

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

environmental process

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

Euler angles

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

event

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

exercise

The DIS term for a one or more interacting simulation applications. Compare to [federation execution](#) in HLA.

exercise connection

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

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

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

federate

A connection to the [RTI](#). Typically a single simulation application can be thought of as a federate.

federation

A group of HLA federates capable of playing in the same [federation execution](#).

Federation Object Model

Defines the data content of a federation execution.

federation execution

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

field of view

Controls the perspective of the observer.

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

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

filter range

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

FOM

See [Federation Object Model](#).

FOM Module

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

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

frame rate

The rate at which the application displays updated images.

ground clamping

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

geocentric coordinates

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

geodetic coordinates

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

gridded data

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

guise

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

head-up display

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

heartbeat

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

High-Level Architecture

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

HLA

See [High-Level Architecture](#).

interaction

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

Logger control PDUs

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

logical range

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

Local RTI Component

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

LRC

See [Local RTI Component](#).

MÄK Technologies Lisp

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

message

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

MIM

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

model definition

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

Modeling and Simulation Coordination Office

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

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See [MÄK Technologies Lisp](#).

orientation clamping

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

object

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

object handle

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

object name

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

observer

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

PDU

See [Protocol Data Unit](#).

Protocol Data Unit

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

radio transmitter

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

radio receiver

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

Realtime Platform Reference FOM

An [HLA](#) reference [FOM](#) based on the [DIS](#) protocol.

recording

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

reflected entity list

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

reference FOM

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

relative dead reckoning

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

remote display engine

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

RID file

See [RTI Initialization Data](#).

RTI Initialization Data

The initialization data required by the RTI for operation.

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

RPR FOM

See [Realtime Platform Reference FOM](#).

RTI

See [Run-Time Infrastructure](#).

Run-Time Infrastructure

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

scene element

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

Simulation Interoperability Standards Organization

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

simulation time

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

SISO

See [Simulation Interoperability Standards Organization](#).

smooth period

The period of time over which [trajectory smoothing](#) takes place.

smoothing

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

state

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

tape

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

terrain following

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

timeout

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

timescale

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

topographic coordinates

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

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

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

UDP port

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

Universal Transverse Mercator

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

UTM

See [Universal Transverse Mercator](#).

view control messages

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



Index

- command-line option 5-24
- appNumber command-line option 4-21, 5-26
- configFile command-line option 5-19, 5-24
- defaultTapeDir command-line option 5-24
- destAddrString command-line option 4-22, 5-25
- deviceAddrString command-line option 4-24, 5-26
- disableTMTTesting command-line option 5-25
- disPort command-line option 4-20, 5-26
- disVersion command-line option 5-26
- enableTrafficAnalyzer command-line option 5-25
- execName command-line option 4-8, 5-25
- exercisId command-line option 4-21, 5-26
- federateName command-line option 4-7, 5-25
- federateType command-line option 4-7, 5-25
- fedFileName command-line option 4-8
- filterEntityIds command-line option 5-26
- filterEntityType command-line option 5-24
- filterObjectNames command-line option 5-25
- fomMapperData command-line option 4-13, 5-25
- fomMapperLib command-line option 4-12, 5-25
- fomModules command-line option 4-9, 5-25
- ignore_rest command-line option 5-24
- ignoreAdvisories command-line option 4-11, 5-25
- logFileName command-line option 5-24
- maxDrainInputTime command-line option 5-25
- mcastAddresses command-line option 4-27, 5-26
- mcastTtl command-line option 4-24, 5-26
- messageLevel command-line option 5-24
- MIM command-line option 4-9, 5-25
- notifyLevel command-line option 5-24
- receiveBufferSize command-line option 4-23
- recvBufferSize command-line option 5-26
- rprFomRevision command-line option 4-11, 5-25
- rprFomVersion command-line option 4-10, 5-25
- runCommand command-line option 5-24
- sendBufferSize command-line option 4-23, 5-26
- showConsole command-line option 5-24
- siteId command-line option 4-21, 5-26
- timeConstrainingRecording command-line option 5-25
- timeRegulatingPlayback command-line option 5-25
- useEmptyFomMapper command-line option 4-12, 5-25
- version command-line option 5-24
- D command-line option 4-24

Numerics

0SISO-REF-010.xml, configuration file 10-12

A

- A command-line option 4-22, 5-25
- a command-line option 4-21, 5-26
- acceleration scaling 3-7
- Access
 - configuring data source for 12-9
 - exporting to 12-8
 - importing data to Excel 12-21
- AddCustomPDU command 13-17

- adding
 - annotation 8-2
 - command to *lgrConfig.xml* 5-22
 - instant annotation 8-3
 - variable to *lgrConfig.xml* 5-22
 - address, multicast 5-26
 - advisory message
 - configuring 13-16
 - ignoring 4-11
 - allowPlaybackWithoutRegions** parameter 5-16, 13-15
 - analyzing, data 10-3
 - annotating, file 8-2
 - annotation 8-2
 - bounding playback to 8-9
 - converting type 8-6
 - creating 8-2
 - deleting 8-6
 - editing 8-5
 - importing annotation file 8-5
 - instant, adding 8-3
 - loading automatically 7-8
 - opening annotation file 8-5
 - saving file 8-4
 - selecting 8-5
 - updating 8-4
 - zooming timeline to 8-8
 - AppendFileToEnd command 13-19
 - appending, file 11-5, 13-19, 13-20
 - application
 - number 4-21, 13-18
 - specifying 5-26
 - argument type 13-5
 - array 10-6, 10-8
 - arrayBracketMap** parameter 10-13
 - asynchronous I/O 13-18
 - configuring 4-22
 - attribute
 - annotation 8-5
 - class 10-5
 - connection 4-4
 - list, hiding 10-3
 - object 10-6
 - scaling, enabling or disabling 13-13
 - updating 10-7
 - viewing object 10-5
 - attribute value update
 - requesting 4-18
 - sending 4-19
 - attributeMap** parameter 10-13
 - automatic
 - playback 7-8
 - recording 7-4
- B**
- backup
 - enabling or disabling 13-7
 - file 7-7, 13-7
 - bak file 7-7, 13-7
 - beginning, moving to 7-15
 - bracket
 - array elements, Traffic Analyzer 10-12
 - broadcast address 4-22, A-2
 - buffer, send and receive 4-23
 - button bar 6-4
 - Bypass Remote Control File, check box 14-5
- C**
- c command-line option 5-24
 - canceling, recording 13-9
 - CancelRecording command 13-9
 - changing
 - connection settings 4-4
 - data delimiter 10-10
 - datatype mapping 10-11
 - delimiter 10-4
 - timebar color 14-7
 - traffic analyzer coordinate system 10-11
 - check box
 - Bypass Remote Control File 14-5
 - Construct File Name 14-3
 - checkpoint, skipping to 7-11
 - checkpoint policy 5-3, 13-9
 - class
 - filtering by (HLA) 9-9
 - interaction 10-5
 - object 10-5
 - classes, subscribing to 4-15
 - clearing
 - file segment 11-3
 - packets from file 13-19
 - ClearSection command 13-19
 - ClosePlaybackFile command 13-7
 - CloseRecordFile command 13-9
 - closing
 - file 6-9, 13-7
 - recording 13-9

-
- color
 - timebar, changing 14-7
 - command 5-2
 - AddCustomPDU 13-17
 - adding to *lgrConfig.xml* 5-22
 - AppendFileToEnd 13-19
 - CancelRecording 13-9
 - ClearSection 13-19
 - ClosePlaybackFile 13-7
 - CloseRecordFile 13-9
 - CommonEntityIds 13-17
 - CommonObjectNames 13-13
 - Connect 13-11
 - ConvertFile 13-19
 - CutSection 13-19
 - DeleteFile 13-6
 - Disconnect 13-11
 - Editor domain 13-19
 - EnableRequestUpdates 4-19, 13-13
 - Exit 13-6
 - ExportFile 13-19
 - ExportSection 13-20
 - FilterEntityIds 13-17
 - FilterEntityTypes 9-3, 13-11
 - FilterObjectNames 13-13
 - Help 13-6
 - HLA simulation domain 13-13
 - in configuration file 5-21
 - list of 13-20
 - LoadPlugin 13-6
 - LoadSubscriptionListFile 4-17, 5-12, 13-13
 - MergeInFile 13-20
 - NewFile 13-9
 - OffsetTime 13-20
 - OpenFile 13-7
 - Pause 13-7
 - PauseRecording 13-9
 - Play 13-7
 - Player domain 13-7
 - PlayPause 13-7
 - PrefixFileToStart 13-20
 - Record 13-9
 - Recorder domain 13-9
 - RecordToMultipleFiles 13-9
 - Restart 13-7
 - RevertFile 13-7
 - running at startup 5-24
 - SaveFile 13-7
 - SaveFileAs 13-7
 - SaveSubscriptionListFile 4-17, 13-14
 - command (continued)
 - SetBackChannelConnection 13-11
 - SetBackupOnSave 7-7, 13-7
 - SetCheckpointParameters 5-4, 13-9
 - SetConnectionParameters 4-6, 4-20, 13-11, 13-14, 13-18
 - SetCurrentTime 13-7
 - SetDdmParameters 5-11, 5-13, 5-15, 13-15
 - SetDefaultTapeDirectory 13-6
 - SetDisableTimeManagementTest 13-16
 - SetDrainInput 13-16, 13-18
 - SetDynamicNamePrefix 3-3, 13-16
 - SetEntityTracking 5-5, 5-6, 13-6
 - SetFastJumpsWithoutDiff 5-5, 5-6, 13-6
 - SetFollowingExportInterval 13-8
 - SetFollowingInterval 13-8
 - SetHeartbeatIntervals 5-5, 5-8, 13-16
 - SetHeartbeatRate 4-25, 13-18
 - SetIgnoreAdvisories 4-11, 13-16
 - SetListenWhenIdle 13-11
 - SetLoopAtEnd 13-8
 - SetNotifyLevel 5-18, 13-6
 - SetPacketViewInterval 13-8
 - SetPassiveSubscription 4-18, 13-17
 - SetPauseOnRecordAs 13-9
 - SetPlaybackReadTimeLimit 5-5, 5-7, 13-6
 - SetPlaybackSpeed 13-8
 - SetPlayOnLoad 13-8
 - SetRecordToMultipleFilesSettings 13-9, 1-2
 - SetRemoteControlId 13-11, 13-12
 - SetRemoteControlSettings 13-12
 - SetSaveConnectionConfiguration 13-13
 - SetScaleAttributes 13-13
 - SetSendOnlyUpdatedValues 4-19, 13-13
 - SetSkippingPlaybackThreshold 13-8, 1-1
 - SetStartStopTimes 13-8
 - SetStopCondition 13-10
 - SetStopDuration 13-10
 - SetStopSize 13-10
 - SetSubscriptionLists 4-16, 5-23, 13-17
 - SetTempDir 13-6
 - SetTimeManagement 4-14, 13-17
 - SetTimeoutIntervals 5-5, 5-9, 13-17
 - SetTimeoutRate 4-26, 13-18
 - SetVRRFFiltering 13-13
 - Simulation domain 13-11
 - SkipTime 13-8
 - StopPlayback 13-8
 - StopRecording 13-10
 - syntax in *lgrConfig.xml* 5-21

- command (continued)
 - System domain 13-6
 - TrimStart 13-20
 - Unpause 13-8
 - UnpauseRecording 13-10
- command-line, Logger Remote Control 14-8
- command-line interface 13-3
 - domains 13-4
- command-line option 5-2
 - 5-24
 - A 4-22, 5-25
 - a 4-21, 5-26
 - appNumber 4-21, 5-26
 - c 5-24
 - configFile 5-19, 5-24
 - D 4-24, 5-26
 - defaultTapeDir 5-24
 - destAddrString 4-22, 5-25
 - deviceAddrString 4-24, 5-26
 - disableTMTTesting 5-25
 - disPort 4-20, 5-26
 - disVersion 5-26
 - e 4-21, 5-26
 - enableTrafficAnalyzer 5-25
 - execName 4-8, 5-25
 - exerciseId 4-21, 5-26
 - F 4-8, 4-10, 5-25
 - f 4-12, 5-25
 - federateName 4-7, 5-25
 - federateType 4-7, 5-25
 - fedFileName 4-8
 - filename 5-24
 - filterEntityIds 5-26
 - filterEntityTypes 5-24
 - filterObjectNames 5-25
 - fomMapperData 4-13, 5-25
 - fomMapperLib 4-12, 5-25
 - fomModules 4-9, 5-25
 - h 5-24
 - ignore_rest 5-24
 - ignoreAdvisories 4-11, 5-25
 - l 5-18, 5-24
 - logFileName 5-24
 - m 5-18, 5-24
 - maxDrainInputTime 5-25
 - mcastAddresses 4-27, 5-26
 - mcastTtl 4-24, 5-26
 - messageLevel 5-24
 - MIM 4-9, 5-25
 - N 4-7, 5-25
- command-line option (continued)
 - n 5-18, 5-24
 - notifyLevel 5-24
 - P 4-20, 5-26
 - r 5-24
 - receiveBufferSize 4-23
 - recvBufferSize 5-26
 - rprFomRevision 4-11, 5-25
 - rprFomVersion 4-10, 5-25
 - runCommand 5-24
 - S 4-27, 5-26
 - s 4-21, 5-26
 - sendBufferSize 4-23, 5-26
 - showConsole 5-24
 - siteId 4-21, 5-26
 - table of 5-24
 - timeConstrainingRecording 5-25
 - timeRegulatingPlayback 5-25
 - useEmptyFomMapper 4-12, 5-25
 - v 5-24
 - version 5-24
 - x 4-8, 5-25
- CommonEntityIds command 13-17
- commonEntityTypes** parameter 9-3
- CommonObjectNames command 13-13
- complex datatype 10-4
- component files 2-3
- concatenating, files 11-5, 13-20
- configuration
 - window, saving and restoring 6-3
- configuration file 5-19
 - OSISO-REF-010.xml 10-12
 - adding string list 5-23
 - datatypeDefault.mtl 10-4
 - entity-type-dict.csv 10-12
 - RTI 2-8
 - specifying 5-24
 - trafficAnalyzerConfig.mtl 10-12
- configuration parameter 5-2
 - useAsyncIO 4-22
- configuring
 - advisory messages 13-16
 - application number 4-21
 - asynchronous I/O 4-22
 - data source for Microsoft Access 12-9
 - data source for MySQL 12-13, 12-15
 - DDM 5-17
 - DIS 4-23
 - heartbeat 13-18
 - timeout 13-18

- configuring (continued)
 - FED file 5-25
 - filtering 13-13
 - high speed playback 7-11
 - HLA connection 13-14
 - indexing 5-3
 - list of common entity types 9-3
 - Logger 5-2
 - for HLA 4-6
 - Logger Remote Control 14-2
 - network read time 5-10
 - passive subscription 13-17
 - performance 5-10
 - playback, fast 5-5
 - Record As 13-9
 - recording 13-10
 - response to Logger Control messages 13-12
 - site ID 4-21
 - system for MÄK RTI 2-8
 - time management 13-17
 - Connect command 13-11
 - connecting 13-11
 - to exercise 4-4
 - to network 4-3
 - connection
 - changing settings 4-4
 - configuring DIS 13-18
 - configuring HLA 13-14
 - editing 4-4
 - Logger Remote Control 14-6
 - properties, saving 4-4
 - settings 4-3
 - status 4-3
 - Connection dialog box 4-3
 - console, displaying 6-2
 - console window 13-3
 - displaying 5-24
 - Construct File Name, check box 14-3
 - control
 - Logger 6-4
 - remote 1-4, 14-2
 - controlling multiple Loggers 1-4, 14-2
 - ConvertFile command 13-19
 - converting, annotation type 8-6
 - conveyed region 5-14
 - coordinate system
 - traffic analyzer, changing 10-11
 - creating
 - annotation 8-2
 - database, MySQL 12-14
 - creating (continued)
 - subscription control list 4-15
 - CSV file 10-12
 - cursor
 - setting to time 13-7
 - timeline 7-15
 - custom PDU, filtering 9-5
 - customPdus** parameter 9-5
 - CutSection command 13-19
 - cutting
 - file segment 11-3
 - packets 13-19
- ## D
- D command-line option 5-26
 - data
 - analyzing 10-3
 - delimiter, changing 10-10
 - exporting 12-2
 - exporting to database 12-3
 - exporting to spreadsheet 12-17
 - importing to Excel 12-21
 - Data Distribution Management 5-11
 - data representation 10-3
 - data source
 - Access 12-9
 - configuring for ODBC 12-6
 - MySQL 12-13, 12-15
 - data type, HLA 12-8
 - Data View 7-16
 - database
 - configuring ODBC data source 12-6
 - exporting data to 12-3
 - exporting HLA data to 12-8
 - importing to Excel 12-21
 - MySQL, creating 12-14
 - schema 12-3
 - datatype 10-3, 10-6, 10-8
 - complex 10-4
 - mapping, changing 10-11
 - supported 10-4
 - datatypeDefault.mtl, configuration file 10-4
 - Datatype Mapper, dialog box 10-11
 - DDM 5-11
 - configuring 5-17
 - conveyed region 5-14
 - playback 5-15
 - subscription list, programming 5-14

- dead reckoning
 - acceleration scaling 3-7
 - velocity scaling 3-7
- dead space, removing 13-20
- dead-reckoning 5-9, 7-8
- DeleteFile command 13-6
- deleting
 - annotation 8-6
 - file segment 11-3
 - packets 13-19
- delimiter
 - changing 10-4
 - changing data 10-10
- Delimiter Editor 10-4
- Delimiter Mapper, dialog box 10-10
- description
 - file
 - when recording 6-12
 - when saving 6-13
- descriptor, region 5-12
- destination address 4-22, 4-23, 13-18
- device address 13-18
- dialog box
 - Connection 4-3
 - Dataype Mapper 10-11
 - Delimiter Mapper 10-10
 - License Setup 2-5
 - Logger File Information 6-10, 6-11
 - New Logger File 6-12
 - Record As 14-3
 - Save File As 6-12
 - Set Start and End Time 7-9
 - Set Zoom 7-14
- directory
 - ./bin* 2-3
 - ./doc* 2-3
 - ./logger_tapes* 2-3
- DIS 1-7
 - application number 4-21
 - asynchronous I/O 4-22
 - configuring 4-23
 - configuring connection 13-18
 - exercise ID 3-3, 4-21
 - export to database 12-7
 - filtering
 - entity types 13-11
 - PDU's 13-17
 - heartbeat 4-25, 13-18

- DIS (continued)
 - port
 - Logger default A-2
 - specifying 5-26, A-2
 - protocol version 5-26
 - site ID 4-21
 - timeout 4-26, 13-18
- DIS Configuration page 4-25, 4-26
- disabling
 - attribute scaling 13-13
 - backup 13-7
 - entity tracking 13-6
 - GUI 6-2
 - high speed playback 7-11
 - looping playback 13-8
 - Pause on Record As 7-4
 - Play On Load 7-8
 - time management 13-16
- Disconnect command 13-11
- disconnecting 13-11
 - from exercise 6-7
- displaying
 - console 6-2
 - console window 5-24
 - file information 6-10
 - FOM tab 10-5
 - object statistics 7-18
 - version information 5-24
- docking, toolbars and plug-in views 6-6
- domain 5-22
 - associated with command 13-20
 - command 13-4
- drain input, maximum 5-25

E

- e command-line option 4-21, 5-26
- editing
 - annotation 8-5
 - connection 4-4
 - file 11-2
 - file information 6-11
 - lgrConfig.xml* 5-19
 - point annotation 8-6
 - range annotation 8-6
 - undoing changes 11-2
- Editor domain, commands 13-19
- empty space, trimming 11-6
- enableDdm** parameter 5-15
- EnableRequestUpdates, command 4-19, 13-13

- enabling
 - attribute scaling 13-13
 - backup 13-7
 - entity tracking 13-6
 - high speed playback 7-11
 - looping playback 13-8
 - Pause on Record As 7-4
 - Play On Load 7-8
 - time management 13-16
- end
 - jumping to range annotation 8-7
 - moving to 7-15
- end time
 - setting dynamically 8-4
 - setting for instant range annotation 8-3
- ending, recording 7-6
- end-of-file, setting 7-10
- entity
 - common types 9-3
 - filtering, entity tracking 5-6
 - filtering by type 13-11
 - maintenance 3-7
- entity ID
 - filtering 5-26
 - filtering by 9-6
- entity tracking 5-6
 - enabling or disabling 13-6
- entity type
 - filtering 13-11
 - filtering by 9-2
 - mapping to string 10-12
 - mapping type to string 10-12
 - removing from filtering list 9-3
- entity-type-dict.csv, configuration file 10-12
- entity-type-map-file** parameter 10-12
- enumeration 10-6, 10-8
- environment variable
 - MAKLMGRD_LICENSE_FILE 2-6
 - RTL_CONFIG 2-8
- example
 - lgrControl 1-6
 - lgrModSqlExporter 1-6
 - loggerApp 1-6
 - matrexTspiFomMapper 1-6
 - plug-in 1-6
 - programs 1-6
 - sampleFilter 1-6
 - stateObserver 1-6
- Excel 12-17
 - importing Access data to 12-21

- exercise
 - connecting to 4-4
 - connection 4-3
 - configuring DIS 13-18
 - disconnecting from 6-7
 - ID 3-3, 4-21, 13-18
 - playback 3-3
 - playing, Logger Remote Control 14-5
 - recording 7-3
 - Logger Remote Control 14-3
- Exit command 13-6
- exiting, Logger Remote Control 14-7
- expanding, playback segment 7-10
- export, log file 12-8
- ExportFile command 13-19
- exporting
 - Access to 12-8
 - data, naming convention 7-5
 - data to database 12-3
 - file 12-2, 13-19
- ExportSection command 13-20

F

- F command-line option 4-8, 4-10, 5-25
- f command-line option 4-12, 5-25
- FDD file 2-8
- features 1-2
- FED file 1-7, 2-8, 4-9
 - changing 3-8
 - specifying 4-8, 4-13, 5-25
- federate, specifying Logger federate name 4-7
- federate type, specifying 4-7, 5-25
- federation execution 1-7
 - name 4-8, 5-25
- file
 - annotating 8-2
 - appending 11-5, 13-19, 13-20
 - backup 7-7, 13-7
 - clearing, packets 13-19
 - clearing segment 11-3
 - closing 6-9, 13-7
 - configuration 5-19
 - CSV 10-12
 - deleting
 - packet 13-19
 - segment 11-3
 - description
 - when recording 6-12
 - when saving 6-13

- file (continued)
 - editing 11-2
 - exporting 12-2, 13-19
 - to database 12-3
 - FED 2-8, 4-13
 - information
 - displaying 6-10
 - editing 6-11
 - jumping to beginning 7-15
 - jumping to end 7-15
 - lgrConfig.xml* 5-2
 - loading 6-7
 - Logger, information 6-10
 - merging 11-5, 13-20
 - offsetting 11-6
 - opening 13-7
 - pausing 7-8
 - playing 7-8
 - playing while recording 7-12
 - rclgr 14-4, 14-5
 - recording to multiple 7-4
 - repairing 6-9
 - reverting to saved version 11-2, 13-7
 - RID 2-8
 - saving 7-7, 13-7
 - segment to new file 7-7
 - to new name 7-7
 - starting 7-3
 - trimming 13-20
 - empty space 11-6
 - upgrading format 6-8
 - VR-Link.fed* 4-8
 - filename, *logger.log* 5-18
 - filename command-line option 5-24
 - filter
 - entity ID 9-6
 - HLA object name 9-7
 - message 1-3
 - Filter By Class dialog box 9-9
 - Filter By Entity Type dialog box 9-2
 - Filter By PDU Kind dialog box 9-4
 - FilterEntityIds command 13-17
 - FilterEntityTypes command 9-3, 13-11
 - filterEntityTypes parameter 9-3
 - filtering (continued)
 - custom PDUs 9-5
 - entities 5-6
 - entity ID 5-26
 - entity type 9-2, 13-11
 - input and output 9-2
 - object names 5-25
 - PDU kind 9-4
 - PDUs 13-17
 - VR-Forces messages 9-10
 - FilterObjectNames command 13-13
 - FLEXlm 2-4
 - FOM 2-7
 - database export 12-8
 - flexibility 3-8
 - primary 4-9
 - supported 1-7
 - FOM Mapper 1-7, 4-12, 5-25
 - initialization data 5-25
 - specifying 4-13
 - specifying at startup 4-10
 - FOM module, specifying 4-9, 5-25
 - FOM tab 10-2
 - displaying or hiding 10-5
 - format, upgrading 6-8
 - frequency, packets for indexing 5-3
 - Frozen bit 5-9
- ## G
- GUI
 - disabling 6-2
 - localizing 2-9
- ## H
- h command-line option 5-24
 - handle, start and end time 7-9
 - heartbeat 13-18
 - defined 3-6
 - HLA 5-8, 13-16
 - setting 4-25
 - Help command 13-6
 - hiding
 - attributes list 10-3
 - FOM tab 10-5
 - high speed, playback 7-11
 - histogram 6-3, 7-16

HLA 1-7
 1516
 playback 3-4
 advisory messages 4-11, 13-16
 commands 13-13
 configuring connection 13-14
 configuring for 4-6
 configuring time management 4-14
 database schema 12-8
 DDM 5-11
 Evolved, playback 3-4
 exporting to database 12-8
 FOM flexibility 3-8
 heartbeat 5-8, 13-16
 object name, filtering by 9-7
 passive subscription 4-18, 13-17
 specifications supported 1-7
 subscription control list 4-15
 subscription list 13-17
 support for 3-8
 time management 5-18
 timeout 5-9, 13-17
 HLA Configuration page 4-14, 4-15, 4-18, 4-19
 HLA Evolved
 exporting to database 12-8
 federate type 4-7
 FOM 4-8
 FOM module 4-9
 MIM 4-9
 HLA Traffic Analyzer 10-2
 enabling and disabling 5-25
 options 10-9
 HLA Traffic Analyzer Settings page 10-9, 10-10, 10-11
 hostname, license server 2-5

I

ID
 entity, filtering by 9-6
 IEEE 1516 1-7
 ignoring, advisory messages 4-11
 immediate
 playback 7-8
 recording 7-3, 7-4
 importing, annotation file 8-5
 incomplete file, repairing 6-9
 indexing, configuring 5-3
 Indexing Settings page 5-4

information
 file
 displaying 6-10
 editing 6-11
 Logger file 6-10
 informational message 5-18
 input, filtering 9-2
 inserting, file 13-20
 installing
 MySQL 12-13
 RTI 2-8
 instant annotation 8-3
 interaction
 class 10-5
 parameter 10-5, 10-8
 viewing 10-5
 passive subscription 4-18
 subscribing to 4-15
 Interactions tab 10-2, 10-7
 interface 1-3
 text 13-3
 interim PDU 3-6, 3-7
 IP address 5-25

J

jumping
 range annotation end 8-7
 range annotation start 8-6
 to beginning of file 7-15
 to end of file 7-15
 to specific time 7-15

K

keyboard, setting annotation times from 8-4

L

-l command-line option 5-18, 5-24
 lgrConfig.xml 4-4, 7-11
lgrConfig.xml 13-3
 configuration file 5-2
 syntax and editing 5-19
 lgrControl example 1-6
 lgrDump 1-6
 lgrModSqlExporter example 1-6
 License Manager 2-4
 license server, hostname 2-5

- License Setup dialog box 2-5
- loading
 - annotation file automatically 7-8
 - configuration file 5-24
 - file 6-7, 13-7
 - subscription list 4-17, 13-13
- LoadPlugin command 13-6
- LoadSubscriptionListFile command 4-17, 5-12, 13-13
- localization 1-3
- localizing, GUI 2-9
- log file 5-18, 5-24
 - sql_export.txt* 12-8
- logFileName parameter 5-18
- Logger
 - controlling multiple 1-4, 14-2
 - recording
 - samples 1-5
 - viewing with Stealth 1-5
 - window 6-3
- Logger command 5-2
- Logger Control
 - configuring 13-12
 - message, configuring response to 13-12
- Logger File Information dialog box 6-10, 6-11
- Logger Preferences dialog box 5-4
- Logger Remote Control 14-2
 - command-line options 14-8
 - configuring Logger for 14-2
 - connection 14-6
 - exiting 14-7
 - playing file 14-5
 - rclgr file extension 14-4, 14-5
 - recording exercise 14-3
 - refresh rate 14-6
 - starting 14-3
- loggerApp example 1-6
- logger.log* 5-24
- loop, enabling or disabling 13-8
- looping, playback 7-10

M

- m command-line option 5-18, 5-24
- macro, file naming convention 7-5
- MAK RTI, configuring for name reservation 3-4
- MÄK RTI, configuring 2-8
- maklanddemo2013 1-5
- MAKLMGRD_LICENSE_FILE, environment variable 2-6

- MAK-RPR2-1-1.fed* file 1-7
- mapping
 - datatype, changing 10-11
 - enumeration to string 10-12
- matrexTspiFomMapper example 1-6
- Max Read Time 5-10
- max-interaction-count parameter 10-8
- MergeInFile command 13-20
- merging, files 11-5, 13-20
- message 13-12
 - advisory 4-11
 - filtering 1-3
 - filtering by object name 9-7
 - filtering VR-Forces 9-10
 - informational 5-18
 - log file 5-24
 - maximum to read per tick 5-10
 - VR-Link 5-18
- Microsoft Access, configuring data source for 12-9
- Microsoft Excel 12-17, 12-21
- MIM, specifying 4-9, 5-25
- Min Read Time 5-10
- MOM and Initialization Module, specifying 4-9
- moving
 - to beginning of file 7-15
 - to beginning or end of playback range 7-15
 - to end of file 7-15
 - to specific time 7-15
- multicast address 4-27, 5-26, 13-18
- multicast time-to-live 4-24, 13-18
 - specifying 5-26
- multiple
 - files, recording to 7-4
 - network card 4-24, 5-26
- multiple Logger, controlling 1-4, 14-2
- MySQL 12-17
 - configuring data source for 12-13, 12-15
 - creating database 12-14
 - installing 12-13

N

- N command-line option 4-7, 5-25
- n command-line option 5-18, 5-24
- name
 - Logger federate 4-7
 - object 10-6
- name reservation 3-4
 - configuring MAK RTI 3-4
- naming, partial files 7-5

- naming convention, for database tables and rows 12-3
 - network
 - connection 4-3
 - read time, configuring 5-10
 - requirements for Logger A-2
 - network card, multiple 4-24, 5-26
 - network device 13-18
 - network interface address 4-23
 - specifying 4-24
 - Network Settings page 4-4, 5-10
 - new file 13-9
 - New Logger File dialog box 6-12
 - NewFile command 13-9
 - notification level, setting 5-18
 - Num PDUs Per Check 5-10
- O**
- object
 - attribute, viewing 10-5
 - class 10-5
 - displaying statistics 7-18
 - name 10-6, 13-16
 - passive subscription 4-18
 - subscribing to 4-15
 - object name
 - filtering 5-25
 - filtering by 9-7
 - Objects list 10-12
 - Objects tab 10-2, 10-6
 - ODBC, configuring data source 12-6
 - OffsetTime command 13-20
 - offsetting
 - file 11-6
 - packets 13-20
 - OMT file 12-8
 - OpenFile command 13-7
 - opening
 - annotation file 8-5
 - file 6-7, 13-7
 - incomplete file 6-9
 - option, HLA Traffic Analyzer 10-9
 - output, filtering 9-2
- P**
- P command-line option 4-20, 5-26
 - packet
 - clearing 11-3
 - packet (continued)
 - count 7-16
 - deleting 11-3
 - from file 13-19
 - dumping to text 12-2
 - offsetting 11-6
 - time 13-20
 - preventing dropped 4-22
 - removing
 - empty 11-6
 - from file 13-19
 - skipping 7-11
 - Packet Count View, scaling 7-17
 - page, Network Settings 4-4
 - parameter
 - `allowPlaybackWithoutRegions` 5-16, 13-15
 - `arrayBracketMap` 10-13
 - `attributeMap` 10-13
 - `commonEntityTypes` 9-3
 - `customPdus` 9-5
 - `enableDdm` 5-15
 - `entity-type-map-file` 10-12
 - `filterEntityTypes` 9-3
 - interaction 10-8
 - `logFileName` 5-18
 - `max-interaction-count` 10-8
 - `playbackDynamicAssociation` 5-16
 - `playbackDynamicExtents` 5-16
 - `recordConveyedRegions` 5-16
 - `requestUpdates` 5-16
 - `showConsole` 13-3
 - `subscribeMode` 5-11
 - `SubscribeRegionsAreGlobal` 5-11
 - `SubscribeRegionsAreSpecified` 5-11, 5-16, 13-16
 - `subscribeUnspecifiedAttributes` 5-13, 5-16, 13-16
 - `useAsyncIO` 4-22
 - viewing interaction 10-5
 - partial file, naming convention 7-5
 - passive subscription 4-18
 - configuring 13-17
 - path, specifying for RTI 2-8
 - Pause command 13-7
 - Pause on Record As 7-3, 7-4
 - PauseRecording command 13-9
 - pausing
 - playback 5-9, 7-8, 13-7
 - recording 7-6, 13-9
 - PDU
 - filtering 13-17
 - filtering by 9-4, 13-17

- PDU (continued)
 - filtering custom 9-5
 - maximum to read per tick 5-10
 - supported for database export 12-7
- performance, configuring 5-10
- Play button 7-8
- Play command 13-7
- Play On Load 6-8, 7-8
- playback 3-3, 6-7
 - automatic 7-8
 - bounding to annotation 8-9
 - DDM 5-15
 - described 3-3
 - exercise ID 3-3
 - fast 5-5
 - faster, configuring 13-6
 - high speed 7-11
 - HLA 1516 3-4
 - HLA Evolved 3-4
 - loading file 6-7
 - looping 7-10, 13-8
 - mode 3-3
 - moving to beginning 7-15
 - moving to end 7-15
 - pausing 5-9, 7-8, 13-7
 - range, moving to beginning or end 7-15
 - segment 7-8, 7-9
 - expanding 7-10
 - setting speed 7-10, 13-8
 - starting 13-7
 - stopping 7-10, 13-8
 - time regulating 5-25
 - time-managed 5-18
 - unpausing 13-8
 - while recording 7-12
- playbackDynamicAssociation** parameter 5-16
- playbackDynamicExtents** parameter 5-16
- playbackMode 5-15, 13-15
- PlaybackRegionNotEnabled 5-15
- PlaybackRegionsUseConveyedExtents 5-15
- PlaybackRegionsUseData 5-15, 13-15
- Player domain, commands 13-7
- playing
 - file 7-8
 - part of a file 7-8, 7-9
- PlayPause command 13-7
- plug-in 6-3
 - docking and undocking views 6-6
 - example 1-6
 - object statistics 7-18

- point annotation
 - adding instantly 8-3
 - converting to range 8-6
 - editing 8-6
- Point Annotation dialog box 8-5
- port 13-18
 - specifying UDP 4-20
- preferences, setting 5-3
- PrefixFileToStart command 13-20
- prefixing, file 13-20
- property
 - annotation 8-5
 - connection 4-3
- protocol
 - supported 1-7
 - version, DIS 5-26

Q

- Qt Linguist 2-9

R

- r command-line option 5-24
- range annotation
 - adding instantly 8-3
 - bounding playback to 8-9
 - converting to point 8-6
 - editing 8-6
 - jumping to end 8-7
 - jumping to start 8-6
 - setting end time 8-4
- Range Annotation dialog box 8-5
- rclgr file extension 14-4, 14-5
- receive buffer 4-23, 13-18
 - size, specifying 5-26
- record, mode 3-3
- Record As, configuring 13-9
- Record As dialog box 14-3
- Record command 13-9
- recordConveyedRegions** parameter 5-16
- Recorder domain, commands 13-9
- recording 3-3
 - and playing 7-12
 - automatically 7-4
 - canceling 13-9
 - closing 13-9
 - configuring 13-10
 - described 3-3

recording (continued)
 exercise 7-3
 ID 3-3
 Logger Remote Control 14-3
 immediately 7-3
 multiple files 7-4
 pausing 7-6, 13-9
 repairing 6-9
 resuming 7-6
 saving 7-7
 starting 7-3
 stopping 7-6
 time constrained 5-25
 time-managed 5-18
 RecordToMultipleFiles command 13-9
 refresh rate, Logger Remote Control 14-6
 region
 conveyed 5-14
 subscription 5-11
 region descriptor 5-12
 region set 5-12
 remote, control 1-4, 14-2
 remote control ID 14-3
 specifying 5-24, 13-11
 Remote Control Logger 1-4
 removing
 dead space 13-20
 empty packets 11-6
 packets from file 13-19
 repairing, incomplete file 6-9
 requesting, attribute value updates 4-18
 requestUpdates parameter 5-16
 resetting, annotation time point 8-4
 Restart command 13-7
 restoring, window configuration 6-3
 resuming, recording 7-6
 RevertFile command 13-7
 reverting
 file 13-7
 to saved file 11-2
 RID file 2-8
 rid.mtl file 2-8
 row, naming conventions 12-3
 RPR FOM 3-8, 4-10
 revision, specifying at startup 4-11
 RTI 2-7
 definition of 2-7
 specifications supported 1-7
 RTI_CONFIG environment variable 2-8

S

-S command-line option 4-27, 5-26
 -s command-line option 4-21, 5-26
 sample recording 1-5
 sampleFilter example 1-6
 Save 7-7
 Save As 7-7
 Save File As dialog box 6-12
 SaveFile command 13-7
 SaveFileAs command 13-7
 SaveSubscriptionListFile command 4-17, 13-14
 saving
 annotation file 8-4
 connection properties 4-4
 file 13-7
 to new name 7-7
 files 7-7
 segment as new file 7-7
 subscription list 4-17, 13-14
 scaling
 Packet Count View 7-17
 time 3-7
 schema, database 12-3
 segment
 playing 7-8, 7-9
 saving as new file 7-7
 setting playback to annotation 8-9
 selecting, annotation 8-5
 send buffer 4-23, 13-18
 size, specifying 5-26
 sending, attribute value update 4-19
 set, region 5-12
 Set Start and End Time dialog box 7-9
 Set Time 3-7, 7-15
 Set Zoom dialog box 7-14
 SetBackChannelConnection command 13-11
 SetBackupOnSave command 7-7, 13-7
 SetCheckpointParameters command 5-4, 13-9
 SetCommonEntityTypes command
 command, SetCommonEntityTypes 13-11
 SetConnectionParameters command 4-6, 4-20,
 13-11, 13-14, 13-18
 SetCurrentTime command 13-7
 SetDdmParameters, command 13-15
 SetDdmParameters command 5-11, 5-13, 5-15
 SetDefaultTapeDirectory command 13-6
 SetDisableTimeManagementTest command 13-
 16
 SetDrainInput command 13-16, 13-18

- SetDynamicNamePrefix command 3-3, 13-16
- SetEntityTracking command 5-5, 5-6, 13-6
- SetFastJumpsWithoutDiff command 5-5, 5-6, 13-6
- SetFollowingExportInterval, command 13-8
- SetFollowingInterval, command 13-8
- SetHeartbeatIntervals, command 5-5, 5-8, 13-16
- SetHeartbeatRate command 4-25, 13-18
- SetIgnoreAdvisories command 4-11, 13-16
- SetListenWhenIdle, command 13-11
- SetLoopAtEnd command 13-8
- SetNotifyLevel command 5-18, 13-6
- SetPacketViewInterval, command 13-8
- SetPassiveSubscription command 4-18, 13-17
- SetPauseOnRecordAs command 13-9
- SetPlaybackReadTimeLimit command 5-5, 5-7, 13-6
- SetPlaybackSpeed command 13-8
- SetPlayOnLoad command 13-8
- SetRecordToMultipleFilesSettings command 13-9, 1-2
- SetRemoteControlId command 13-11, 13-12
- SetRemoteControlSettings command 13-12
- SetSaveConnectionConfiguration, command 13-13
- SetScaleAttributes command 13-13
- SetSendOnlyUpdatedValues, command 4-19, 13-13
- SetSkippingPlaybackThreshold command 13-8, 1-1
- SetStartStopTimes command 13-8
- SetStopCondition command 13-10
- SetStopDuration command 13-10
- SetStopSize command 13-10
- SetSubscriptionLists command 4-16, 5-23, 13-17
- SetTempDir command 13-6
- SetTimeManagement command 4-14, 13-17
- SetTimeoutIntervals, command 5-5, 5-9, 13-17
- SetTimeoutRate command 4-26, 13-18
- setting
 - checkpoint policy 13-9
 - cursor to time 13-7
 - DIS heartbeat 4-25
 - DIS timeout 4-26
 - end-of-file action 7-10
 - notification level 5-18
 - playback segment to annotation 8-9
 - playback speed 7-10, 13-8
 - preferences 5-3
- SetVRFFiltering command 13-13
- showConsole parameter 13-3
- simulation
 - recording 7-3
 - standard, supported 1-7
- Simulation domain, commands 13-11
- SISO 1516 API 1-7
- SISO enumeration XML document 10-12
- site ID 4-21, 5-24, 13-11, 13-18
 - specifying 5-26
- size, indexing 5-3
- skip command, entity maintenance 3-7
- Skip Time 3-7
- skipping, packets 7-11
- skippingPlaybackThreshold 7-11
- SkipTime command 13-8
- slider
 - start and end time 7-9
 - zoom 7-9
- slow-speed playback 3-6
- specifying
 - application number 5-26
 - configuration file 5-24
 - FED file 4-8, 4-13, 5-25
 - federate type 4-7, 5-25
 - federation execution name 4-8
 - FOM Mapper 4-13
 - FOM modules 4-9, 5-25
 - MIM 4-9, 5-25
 - MOM and Initialization Module 4-9
 - multicast time-to-live 5-26
 - network interface address 4-24
 - receive buffer size 5-26
 - remote control ID 5-24, 13-11
 - send buffer size 5-26
 - site ID 5-26
 - subscription list 13-17
 - subscription region 5-12
 - time management 5-18
 - UDP port 4-20
- speed
 - control 6-4
 - setting for playback 7-10
 - setting playback 13-8
- spreadsheet, exporting data to 12-17
- SQL, exporting data to 12-3
- sql_export.txt log file 12-8
- start
 - jumping to range annotation 8-6
 - moving to 7-15
- start time, setting dynamically 8-4

-
- starting
 - Logger 6-2
 - Logger Remote Control 14-3
 - playback 13-7
 - recording 7-3
 - startup
 - playback mode 3-3
 - record mode 3-3
 - running commands 5-24
 - UNIX 6-2
 - Windows 6-2
 - stateObserver example 1-6
 - status, field 7-3
 - stopping
 - playback 7-10, 13-8
 - recording 7-6
 - StopPlayback command 13-8
 - StopRecording command 13-10
 - string list 4-16
 - adding to configuration file 5-23
 - subnet mask 4-23, 13-18
 - subscribeMode 5-15
 - subscribeMode** parameter 5-11
 - SubscribeRegionNotEnabled 5-15, 13-15
 - SubscribeRegionsAreGlobal 5-15
 - SubscribeRegionsAreGlobal** parameter 5-11
 - SubscribeRegionsAreSpecified 5-15
 - SubscribeRegionsAreSpecified** parameter 5-11, 5-16, 13-16
 - subscribeUnspecifiedAttributes** parameter 5-13, 5-16, 13-16
 - subscribing, passively 4-18
 - subscription, region 5-11
 - subscription control list, creating 4-15
 - subscription list
 - DDM, setting programmatically 5-14
 - loading 4-17, 13-13
 - saving 4-17, 13-14
 - specifying 13-17
 - subscription region, specifying 5-12
 - syntax, *lgrConfig.xml* 5-19
 - System domain, commands 13-6
 - TCP 5-26
 - temporary file, recording to 7-3
 - text, dumping file to 12-2
 - text interface 13-3
 - time
 - command, entity maintenance 3-7
 - indexing 5-3
 - jumping to 7-15
 - navigating during playback 7-15
 - navigation 7-15
 - offsetting packet 13-20
 - setting 7-15
 - setting cursor to 13-7
 - sliders 7-9
 - time constrained, recording 5-25
 - time jump, updating entities after 3-7
 - time management
 - configuring 13-17
 - in GUI 4-14
 - enabling or disabling 13-16
 - specifying use of 5-18
 - time regulating, playback 5-25
 - time scale
 - described 3-4, 3-5, 3-6
 - during recording 7-3
 - effects 3-6
 - timebar
 - color, changing 14-7
 - time-constrained, setting 4-14
 - timeline
 - cursor 7-15
 - moving 7-15
 - zooming 7-13
 - to annotation 8-8
 - timeout 13-18
 - defined 3-6
 - HLA 5-9, 13-17
 - setting 4-26
 - timepoint, setting 7-15
 - time-regulating, setting 4-14
 - timestamp, offsetting 11-6
 - Toggle Attributes Pane bar 10-3
 - toolbar 6-4
 - docking and undocking 6-6
 - Traffic Analyzer 10-2
 - array elements, brackets 10-12
 - trafficAnalyzerConfig.mtl, configuration file 10-12
 - translating, GUI text 2-9
 - translation 1-3
- T**
- table, naming conventions 12-3
 - tape
 - annotating 8-2
 - editing 11-2
 - loading 6-7

trimming, file 11-6, 13-20
TrimStart command 13-20
type, object 10-6

U

UDP port 5-26
 specifying 4-20, 5-26
undo, edits 11-2
undocking, toolbars and plug-in views 6-6
UNIX
 starting, Logger 6-2
Unpause command 13-8
UnpauseRecording command 13-10
unpausing 13-8
Update Annotation 8-4
updating, annotation 8-4
Upgrade File Format 6-8
upgrading, file format 6-8
useAsyncIO parameter 4-22

V

-v command-line option 5-24
variable
 adding to *lgrConfig.xml* 5-22
 syntax in *lgrConfig.xml* 5-21
variant 10-6, 10-8
VCR controls 6-4
velocity, scaling 3-7
version, displaying 5-24
View Annotations window 8-2
view control PDUs 1-5
View menu 6-5
viewing
 connection properties 4-3
 interaction parameter 10-5
 object attribute 10-5
 object statistics 7-18
VR-Forces message 9-10
VR-Link, messages 5-18
VR-Link.fed file 4-8
VR-TheWorld_FirstExperienceDIS 1-5

W

window
 configuration 6-3
 Logger 6-3

Windows
 installing on 2-2
 starting Logger 6-2

X-Y-Z

-x command-line option 4-8, 5-25
XML, RTI configuration file 2-8
XML file 12-8
zoom area 7-9
zooming
 Packet Count View 7-17
 timeline 7-13
 timeline to annotation 8-8

MÄK Data Logger Quick Reference Card

Logger commands

Command	Description
System domain commands	
DeleteFile <i>filename</i> (string)	Deletes the specified file.
Exit	Shuts down the Logger.
Help	Displays a list of available commands.
LoadPlugin <i>pluginName</i> (string)	Loads the specified plug-in.
SetDefaultTapeDirectory <i>defaultTapeDir</i> (string)	Sets the directory where the Logger creates and loads files.
SetEntityTracking <i>entityTracking</i> (bool)	Allows entity tracking to be processed.
SetFastJumpsWithoutDiff <i>fastJumpsWithoutDiff</i> (bool)	Uses checkpoints to establish the relative state of objects, but the packets between the checkpoint and jump time are not processed
SetNotifyLevel <i>vlLevel</i> (int) <i>lgrLevel</i> (int)	Sets the notification level of VR-Link and the Logger.
SetPlaybackReadTimeLimit <i>playbackReadTimeLimit</i> (double)	Limits how long the Logger can spend reading packets during playback.
SetTempDir <i>directory</i> (string)	Sets the directory in which the Logger stores temporary Logger files.
Player domain commands	
ClosePlaybackFile	Closes the current playback file.
OpenFile <i>filename</i> (string) <i>constructFilename</i> (bool)	Opens the specified file for playback.
Pause	Pauses playback.
Play	Starts playback. If the Logger is already playing this command has no effect.
PlayPause	Toggles between the paused and play state.
Restart	Starts playing from the beginning of the file.
RevertFile	Reverts a previously saved, but currently unsaved file to the previously saved state.
SaveFile <i>reopenAfterSave</i> (bool)	Saves a previously saved, but currently unsaved file to its current file name.
SaveFileAs <i>filename</i> (string) <i>description</i> (string) <i>writeDescription</i> (bool) <i>reopenAfterSave</i> (bool)	Saves the current file to new file.
SetBackupOnSave <i>backup</i> (bool)	Enables or disables automatic backup of previously saved files when saving edited files.
SetCurrentTime <i>time</i> (time)	Moves the timeline cursor to the specified time.
SetFollowingInterval	Specifies the default length of the zoom timeline, in seconds
SetFollowingExportInterval	Specifies the length of timeline data export frequency when playing Logger data from an active recording.
SetLoopAtEnd <i>value</i> (bool)	Enables or disables looping to start at the end of playback.
SetPacketViewInterval <i>interval</i> (double)	Sets the interval between tick marks in the Packet Count View and Packet Size View.
SetPlaybackSpeed <i>speed</i> (double) <i>skippingPlayback</i> (bool)	Sets the playback speed of the Logger.
SetPlayOnLoad <i>value</i> (bool)	Enables starting playback immediately after opening a file.
SetSkippingPlaybackThreshold <i>skippingPlaybackThreshold</i> (double)	Set the playback speed above which skipping playback mode is used automatically.
SetStartStopTimes <i>startTime</i> (time) <i>stopTime</i> (time)	Sets the start and stop time of the Logger.

Logger commands

Command	Description
SkipTime	Move the timeline cursor (forward or backward) relative to current time.
StopPlayback	Stops playback. This command has no effect if the Logger is not in the play-back state.
Unpause	Resumes playback. This command has no effect if the Logger is not paused.
Recorder domain commands	
CancelRecording	Cancels the current recording.
CloseRecordFile	Closes the file currently recording.
NewFile <i>filename</i> (string) <i>description</i> (string) <i>constructFilePathAndName</i> (bool) <i>overwrite</i> (bool)	Opens a new file with the given name and description for recording.
PauseRecording	Pauses the recording process.
Record	Starts recording to a new file.
RecordToMultipleFiles <i>fileName</i> (string)	Starts recording to new files.
SetCheckpointParameters <i>policy</i> (int) { <i>size</i> (int) <i>count</i> (int) <i>time</i> (time)}	Sets the checkpoint policy parameters for generating checkpoints. ♦ 0 — count, in number of packets. ♦ 1 — size of packets, in bytes. ♦ 2 — time, in seconds between checkpoints.
SetPauseOnRecordAs <i>record</i> (bool)	Determines how the Logger behaves when given the Record As command.
SetRecordToMultipleFilesSettings <i>duration</i> (double) <i>namingConvention</i> (string)	Set the multiple file recording parameters.
SetStopCondition <i>condition</i> (int)	Specifies the stop condition for recording as follows: ♦ 0 — no condition. ♦ 1 — duration (enables SetStopDuration). ♦ 2 — file size (enables SetStopSize).
SetStopDuration <i>duration</i> (time)	Stops recording at the specified time if SetStopCondition is set to 1.
SetStopSize <i>filesize</i> (int64)	Stops recording when the file reaches <i>filesize</i> if SetStopCondition is 2.
StopRecording <i>closeAndOpenForPlayback</i> (bool)	Stops recording and ends.
UnpauseRecording	Resume a paused recording.
Simulation domain commands (protocol-independent commands)	
Connect	Connects to the network simulation using the previously set parameters.
Disconnect	Disconnects from the network simulation.
FilterEntityTypes <i>entityType1</i> ... <i>entityTypeN</i> (string)	Specifies a list of entity types to list in the Filter by Entity Type dialog box and enables filtering for these entity types.
SetBackChannelConnection <i>useBackChannel</i> (bool) <i>address</i> (string) <i>port</i> (int) <i>interface</i> (string)	Set the parameters for the back channel connection configuration.
SetCommonEntityTypes <i>entityType1</i> ... <i>entityTypeN</i> (string)	Specifies a list of entity types to list in the Filter by Entity Type dialog box.
SetListenWhenIdle <i>state</i> (bool)	Enable or disable listening to the network when not recording or playing.
SetRemoteControlSettings <i>process</i> (bool) <i>process</i> (bool) <i>recordskippingPlayback</i> (bool) <i>allowDelete</i> (bool) <i>allowAll</i> (bool) <i>sendResponses</i> (bool) <i>crippleGui</i> (bool) <i>useLegacyPdu</i> (bool) <i>connectSendInterval</i> (double) <i>heartbeatSendInterval</i> (double)	Specifies how the Logger responds to Logger Control messages.
SetSaveConnectionConfiguration <i>saveConnectionConfiguration</i> (bool)	Enables or disables saving of connection information to <i>lgrConfig.xml</i> .
SetScaleAttributes <i>scale</i> (bool)	Enables or disables attribute scaling due to changes in playback speed.

MÄK Data Logger Quick Reference Card

Logger commands

Command	Description
<code>SetSendOnlyUpdatedValues state (bool)</code>	If true, when the Logger receives attribute value update requests, only send updates for values that have been updated in the Logger file. Do not send default values.
<code>SetVRFFiltering filter(bool) loId(int) hiId(int)</code>	Configures VR-Forces message filtering.
HLA-specific commands	
<code>CommonObjectNames objectName1 ... objectNameN(string)</code>	Specifies a list of object names to list in the Filter by Object Names dialog box. Does not enable filtering of these object names.
<code>EnableRequestUpdates state (bool)</code>	Enable or disable whether request updates is invoked when an object is discovered. Default: true.
<code>FilterObjectNames objectName1 ... objectNameN(string)</code>	Specifies a list of object names to list in the Filter by Object Names dialog box and enables filtering for these entity object names.
<code>LoadSubscriptionListFile filename(string)</code>	Loads a subscription list file.
<code>SaveSubscriptionListFile filename(string)</code>	Saves the subscription settings to an XML file with the specified name.
<code>SetConnectionParametersHLA federateType(string) federationName(string) fedFile(string) ignoreAdvisories(bool) fomVersion(double) fomRevision(int) fomMapper(string) fomMapperData(base64) useEmptyFomMapper(bool) useMatrexFomMapper(bool) matrexFomMapperData(string)</code>	
<code>SetConnectionParametersHLAe federateName(string) federateType(string) federationName(string) fedFile(string) fomModules(string) MIM(string) ignoreAdvisories(bool) fomVersion(double) fomRevision(int) fomMapper(string) fomMapperData(base64) useEmptyFomMapper(bool) useMatrexFomMapper(bool) matrexFomMapperData(string)</code>	Specifies HLA (1.3 and 1516) or HLA Evolved connection parameters.
<code>SetDdmParameters enableDdm playbackMode subscribeMode playbackDynamicAssociation playbackDynamicExtents allowPlaybackWithoutRegions recordConveyedRegions requestUpdates subscribeUnspecifiedAttributes</code>	Specifies DDM parameters.
<code>SetDisableTimeManagementTest disableTesting(bool)</code>	Disables time management testing if it causes the Logger to misbehave.
<code>SetDrainInput maxTime(double), minTime(double)</code>	Specifies the minimum and maximum amount of time, in seconds, to spend reading messages from the network each tick.
<code>SetDynamicNamePrefix prefix (string)</code>	Specifies the prefix that the Logger attaches to the beginning of object names that begin with HLA_ when it is recording exercises.
<code>SetHeartbeatIntervals "className(string):interval(float):inheritable(bool) ..."</code>	Specifies a heartbeat interval for playback. You can specify multiple classes.
<code>SetIgnoreAdvisories mode(bool)</code>	Specifies the Logger's response to HLA advisory messages. Default: false.
<code>SetPassiveSubscription mode(bool)</code>	Configures passive subscription for HLA. Default: false.
<code>SetSubscriptionLists objectClasses(stringList) interactionClasses(stringList)</code>	Sets the class subscription list.
<code>SetTimeManagement timeConstrainedRecording(bool) timeConstrainedPlayback(bool) timeRegulatingRecording(bool) timeRegulatingPlayback(bool) lookahead(double)</code>	Configures time management options.

Logger commands

Command	Description
<code>SetTimeoutIntervals "className(string):interval(float):inheritable(bool) ..."</code>	Specifies an object timeout for recording. You can specify multiple classes.
DIS-specific commands	
<code>AddCustomPDU pduName(string) pduKind(string) entityIdOffset(string) entityTypeOffset(string)</code>	Specifies a PDU to add to the Filter by PDU Kind dialog box.
<code>CommonEntityIds id1 ... idN(string)</code>	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box. Does not enable filtering of these entity IDs.
<code>FilterEntityIds id1 ... idN(string)</code>	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box and enables filtering for these entity IDs.
<code>SetConnectionParameters port(int) exerciseId(int) siteId(int) appNumber(int) useAsynchIO(bool) sendBufferSize(int) recvBufferSize(int) multicastTtl(int) destAddr(string) subnetMask(string) deviceAddr(string) mcastAddresses(stringList)</code>	Specifies parameters for the DIS exercise connection.
<code>SetDrainInput maxTime(double), numPdusToRead(int)</code>	Specifies the maximum amount of time, in seconds, to spend reading messages from the network and the number of PDUs to read between each check of the maximum time.
<code>SetHeartbeatRate rate(double)</code>	Specifies the DIS heartbeat rate.
<code>SetTimeoutRate rate(double)</code>	Specifies the DIS timeout rate.
Editor domain commands	
<code>AppendFileToEnd filename(string)</code>	Appends the file <i>filename</i> to the end of the open file.
<code>ClearSection start(time) end(time)</code>	Removes all packets between the <i>start</i> and <i>end</i> times in a file.
<code>ConvertFile</code>	Converts the current file to the latest file format.
<code>CutSection start(time) end(time)</code>	Removes all packets between the <i>start</i> and <i>end</i> times in a file.
<code>ExportFile exportFormat(string) destination(string) checkpoint(bool) openAfterExport(bool)</code>	Exports a file using a given <i>exportFormat</i> to a <i>destination</i> .
<code>ExportSection exportFormat(string) destination(string) start(time) end(time) offset(double) checkpoint(bool) openAfterExport(bool)</code>	Exports a section of a file using a given <i>exportFormat</i> to a <i>destination</i> .
<code>MergeInFile filename(string) offset(double)</code>	Merges the Logger file <i>filename</i> into the open file.
<code>OffsetTime offset(double)</code>	Offsets the time of the packets in the open file by the <i>offset</i> value.
<code>PrefixFileToStart filename(string)</code>	Inserts the file named <i>filename</i> at the start of the open file.
<code>TrimStart</code>	Removes any dead time from the start of the file and offsets all packets so that the first packet time stamp is 0.0.

MÄK Data Logger Quick Reference Card

Keyboard Shortcuts

Action	Key	Action	Key
Annotations	Shift + A	Open	Ctrl + O
Append to End	Ctrl + Shift + E	Pause	Ctrl + P
Clear Section	Ctrl + Shift + Delete	Packet Count View	Shift + C
Close	Ctrl + W	Packet Size View	Shift + S
Connection	Ctrl + Shift + C	Play	Ctrl + P
Create Instant Point Annotation	\	Preferences	Ctrl + Shift + P
Create Instant Range Annotation	[	Prefix to Start	Ctrl + Shift + W
Create Point Annotation	Ctrl + \	Properties	Return
Create Range Annotation	Ctrl + [	Record	Ctrl + R
Decrease Speed	Ctrl + Down	Record As	Ctrl + N
Delete Annotation	Delete	Record To Multiple Files	Ctrl + M
Delete Section	Ctrl + X	Revert	Ctrl + Shift + R
Disconnect	Ctrl + D	Save	Ctrl + Shift + S
Exit	Ctrl + Q	Save As	Ctrl + Shift + A
Increase Speed	Ctrl + Up	Save Recorded Data	Ctrl + Shift + S
Jump to End	End	Set Start and End Times	Ctrl + B
Jump to Start	Home	Set Zoom	Ctrl + I
Jump to Time	Ctrl + T	Stop	Ctrl + S
Merge In File	Ctrl + Shift + I	Toggle Loop at End	Ctrl + L
Modify Annotation	Return	Trim Start	Ctrl + Shift + T
Object Statistics	Shift + O	Update Annotation	]
Offset Time	Ctrl + Shift + O	Upgrade File Format	Ctrl + Shift + U

Command-line Options

Option	Description
<code>--allowAllRemoteControl mode</code>	Allow (1) or do not allow (0) remote control PDUs.
<code>-c mode</code>	If <i>mode</i> = 1, show the Logger console.
<code>--configFile filename</code>	Specifies the configuration file to load.
<code>--defaultTapeDir directory</code>	Specifies the default directory for Logger files.
<code>--filterEntityTypes</code>	Adds entity types to Filter by Entity Type dialog box and selects them.
<code>--ignore_rest</code>	Ignore all arguments after this one.
<code>-h</code>	Displays a brief summary of Logger command-line arguments.
<code>-l logfile</code>	Specifies a file to send notification messages to. Default: <i>logger.log</i> .
<code>--loggerTickRate rate</code>	The interval for the Logger tick rate, in milliseconds.
<code>-m level</code>	Specifies the notification level for Logger messages. Range 0-4.
<code>-n level</code>	Specifies the notification level for VR-Link messages. Range 0-4.
<code>-r command</code>	Specifies a Logger command to run. Can be used multiple times.
<code>--remoteControlId ID</code>	Specifies the Logger's remote control ID. Default: site ID.
<code>-v</code>	Displays version information and exits.

HLA Only

<code>--disableTMTesting mode</code>	Disable time management for testing.
<code>--enableTrafficAnalyzer mode</code>	Enable (1) or disable (0) the HLA Traffic Analyzer.
<code>-f libname</code>	Specifies the FOM Mapper to use if you are not using the default FOM.
<code>--federateType type</code>	Specifies the federate type for the Logger federate.
<code>--filterObjectNames names</code>	Filter the listed object names. Accepted multiple times.
<code>--fomMapperData data</code>	Specifies FOM Mapper initialization data, if required.
<code>--fomModules "module1 module2 ... moduleN"</code>	Specifies one or more FOM modules for the exercise. (HLA Evolved only)
<code>-F fedfile</code>	Specifies the FED file for the exercise.
<code>--ignoreAdvisories mode</code>	If 1, ignore HLA advisory messages. If 0, do not ignore.
<code>--maxDrainInputTime time</code>	Specifies the maximum time to drain input.
<code>--MIM MIM_file</code>	Specifies the MIM for the exercise. (HLA Evolved only)
<code>-N federate_name</code>	Specifies the federate name for the Logger federate. (HLA Evolved only)
<code>--rprFomVersion version</code>	Specifies the version of the RPR FOM to use.
<code>--rprFomRevision revision</code>	Specifies the FOM revision to use.
<code>--ignoreAdvisories {0 1}</code>	Specifies whether (1) or not (0) the Logger ignores advisory messages.
<code>--timeConstrainingRecording mode</code>	Use time constrained for recording.
<code>--timeRegulatingPlayback mode</code>	Use time regulating for playback
<code>--useEmptyFomMapper {0 1}</code>	Specifies whether (1) or not (0) the Logger uses an empty FOM Mapper.
<code>-x ExName</code>	Specifies the federation execution name. Default: VR-Link.

DIS Only

<code>-A address</code>	Specifies the default IP address to which the Logger sends PDUs. If this option is not used, the Logger uses the computer's broadcast address.
<code>-a appnumber</code>	Specifies the application number.

Command-line Options

Option	Description
(-D --deviceAddString) <i>address</i>	Specifies the network interface address.
--disVersion <i>num</i>	Sets the DIS protocol version for PDUs sent onto the network. <i>num</i> can be 4 (DIS 2.0.4), 5 (IEEE 1278.1), or 6 (DIS 2.1.4, IEEE 1278.1a).
(-e --exerciseId) <i>ID</i>	Sets the exercise ID.
--filterEntityIds <i>IDs</i>	Filter the specified entity IDs. Acceptable multiple times.
--mcastTtl <i>tTl</i>	Specifies the multicast time-to-live.
-P <i>port</i>	Specifies which DIS port (UDP port if -A not specified, otherwise, TCP) the Logger uses to send or receive data. Default: 3000.
--recvBufferSize <i>size</i>	Specifies the receive buffer size.
--sendBufferSize <i>size</i>	Specifies the send buffer size.
-s <i>ID</i>	Specifies the site ID.



Link - Simulate - Visualize

LOG-5.4-1-160325