MAK TECHNOLOGIES

MAK Data Logger Users Guide

Version 5.9

ST Engineering North America

Copyright © 2021 MAK Technologies
All rights Reserved. Printed in the United States.

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

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

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

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

SilverLining™ is a trademark of Sundog Software.

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

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

All other trademarks are owned by their respective companies.

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

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

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

info@mak.com

www.mak.com

Revision LOG-5.9-1-210812

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 xix
libXML and libICONV xix
QScintilla and SciTE xix

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 1-3
1.1.5. Streaming Video 1-4
1.1.6. Data Export 1-4
1.1.7. HLA Traffic Analyzer 1-4
1.1.8. The Remote Control Logger 1-4
1.1.9. Sample Logger Files 1-5
1.1.10. 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 MAK Data Logger 2-2
2.1.1. Installing MAK Data Logger on Windows 2-2
2.1.2. Installing Logger on a Linux System 2-3
2.2. The Logger Directory Structure 2-4
2.3. Installing and Setting Up the MAK License Manager 2-5
2.3.1. Specifying the License Server 2-6
2.4. Installing an RTI 2-8
2.4.1. Installing the MAK RTI 2-8
2.5. Localizing the Graphical User Interface 2-10
2.5.1. Translating Other Interface Files 2-13
2.5.2. Applying the Language Files 2-14

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 4-10
4.2.9. Ignoring Advisory Messages 4-11
4.2.10. Specifying a FOM Mapper 4-11
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 4-17
4.2.14. Enabling Request Attribute Updates 4-18
4.2.15. Sending Only Updated Values 4-18

4.3. Configuring the Logger for DIS 4-19
4.3.1. Specifying a UDP Port 4-20
4.3.2. Setting the Exercise ID 4-20
4.3.3. Setting the Site ID 4-20
4.3.4. Setting the Application Number 4-21
4.3.5. Using Asynchronous I/O 4-21
4.3.6. Specifying a Destination Address for Outgoing and Incoming Data 4-21
4.3.7. Setting the Subnet Mask 4-22
4.3.8. Setting the Buffer Size for Sending and Receiving PDUs 4-22
4.3.9. Setting the Multicast Time to Live Value 4-22
4.3.10. Setting the Network Interface Address 4-23
4.3.11. Specifying the Heartbeat 4-24
4.3.12. Specifying the Timeout 4-25
4.3.13. Specifying the DIS Version 4-25
4.3.14. Subscribing to Multicast Groups 4-26

Chapter 5. Configuring the Logger

5.1. Configuring the Logger 5-2
5.2. Configuring Indexing 5-2
5.2.1. Configuring Indexing Using the Command Interfaces 5-4
5.3. Configuring Faster Playback 5-4
5.3.1. The SetEntityTracking Command 5-5
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 *lgrConfig.xml* 5-18
5.7. Setting the Notification Level 5-19
5.7.1. Specifying the Log File 5-19
5.8. Configuring Streaming Video 5-20
5.8.1. Video Record Settings 5-20
5.8.2. Video Playback Settings 5-21
5.8.3. Video Advanced Settings (Configuring GStreamer) 5-22
5.9. Editing lgrConfig.xml 5-23
5.9.1. Adding a Command to *lgrConfig.xml* 5-26
5.10. Summary of Command-line Options for the Logger 5-28

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 6-4
6.2.3. Docking and Undocking GUI Plug-in Views and Toolbars 6-5
6.2.4. Saving the Window Layout 6-6
6.3. Connecting to the Simulation Network 6-6
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 6-9
6.4.2. Opening Incomplete Files 6-10
6.4.3. Closing a Logger File 6-10
6.5. Listening to the Network 6-11
6.6. Viewing and Editing Logger File Information 6-11
6.6.1. Displaying Information about a Logger File 6-12
6.6.2. Editing the Logger File Description 6-12

Chapter 7. Recording Logger Files

7.1. Recording Exercises 7-2
7.1.1. Recording Immediately (to a Temporary File) 7-2
7.1.2. Recording to a Specified File 7-2
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. Enabling Video Recording 7-7
7.2.1. Recording Streaming Video from VR-Vantage 7-7
7.2.2. Recording Streaming Video from a Webcam 7-8
7.3. Saving Files 7-9
7.3.1. File Backups 7-9
7.3.2. Saving a File to a New Name 7-9
7.3.3. Saving a Portion of a File to a New File 7-10

Chapter 8. Playing Logger Files

8.1. Playing Logger Files 8-2
8.1.1. Playing a Logger File 8-2
8.1.2. Pausing Playback of a Logger File 8-2
8.1.3. Looping Playback 8-2
8.1.4. Stopping Playback of a Logger File 8-3
8.1.5. Viewing Video 8-3
8.1.6. Playing a Segment of a File 8-3
8.1.7. Setting the Playback Speed 8-6
8.1.8. Using High Speed Playback 8-6
8.1.9. Playing a File While It Is Being Recorded 8-7

8.2. Zooming the Timeline 8-9
8.2.1. Zooming the Timeline Using the Mouse 8-9
8.2.2. Setting the Zoom from the Set Zoom Dialog Box 8-10
8.2.3. Moving the Zoomed Selection 8-10
8.3. Moving the Timepoint to a Specific Time 8-11
8.3.1. Jumping to the Beginning or End of the Playback Range 8-11
8.3.2. Moving to a Specific Time in a File 8-11
8.3.3. Setting the Time Using the Timeline Cursor 8-11
8.4. Viewing Packet Histograms 8-12
8.4.1. Scaling the Packet Count View 8-13
8.5. Viewing Object Statistics 8-13
8.6. Viewing Streaming Video Using VLC 8-14
8.6.1. Reducing Latency in VLC 8-14

Chapter 9. Annotating Files

9.1. Annotating a Logger File 9-2
9.1.1. Adding an Annotation 9-2
9.1.2. Saving an Annotation File 9-4
9.1.3. Opening an Annotation File 9-5
9.1.4. Importing an Annotation File 9-5
9.1.5. Selecting an Annotation 9-5
9.1.6. Editing an Annotation 9-5
9.1.7. Deleting an Annotation 9-6
9.1.8. Moving the Timeline Cursor to the Start of an Annotation 9-7
9.1.9. Jumping to the Next or Previous Annotation 9-7
9.1.10. Zooming the Timeline to an Annotation 9-8
9.1.11. Bounding Playback to an Annotation 9-9

Chapter 10. Filtering Input and Output

10.1. Filtering Logger Input and Output 10-2
10.2. Filtering by Entity Type 10-3
10.2.1. Removing Entity Types from the Entity Types Lists 10-3
10.2.2. Configuring the List of Common Entity Types 10-4
10.3. Filtering DIS PDUs by PDU Kind 10-5
10.4. Including Custom PDUs in Filter Lists 10-6
10.5. Filtering DIS Messages by Entity ID 10-7
10.6. Filtering HLA Messages by Object Name 10-8
10.7. Filtering HLA Exercises by Class 10-10
10.8. Filtering by Marking Text 10-11
10.9. Filtering VR-Forces Messages 10-12

Chapter 11. Analyzing HLA Data

11.1. The HLA Traffic Analyzer 11-2
11.2. Traffic Analyzer Data Views 11-2
11.2.1. Hiding the Attributes Pane 11-3

11.3. How the Traffic Analyzer Evaluates Data 11-3
11.3.1. Representing Complex Datatypes 11-4
11.3.2. Datatype Representations Included in the Traffic Analyzer 11-4
11.4. Viewing Object and Interaction Classes and Attributes 11-5
11.4.1. Viewing the Attributes of an Object Class 11-5
11.4.2. Viewing the Parameters of an Interaction Class 11-5
11.5. Viewing Object Instances 11-6
11.6. Viewing Interaction Instances 11-7
11.7. Configuring the HLA Traffic Analyzer 11-9
11.7.1. Changing the Data Delimiter 11-10
11.7.2. Mapping Datatypes 11-11
11.7.3. Configuring Attribute Display Options 11-12
11.7.4. Mapping Entity Type Enumerations to Strings 11-12
11.7.5. Removing Brackets from the Display of Array Types 11-13

Chapter 12. Editing Logger Files

12.1. Editing Files 12-2
12.1.1. Reverting to the Saved Version of a File 12-2
12.2. Clearing and Deleting Segments of a File 12-3
12.3. Merging Files 12-4
12.4. Concatenating Files 12-5
12.5. Trimming Files 12-5
12.6. Offsetting Files 12-5

Chapter 13. Exporting Logger Data

13.1. Exporting Data 13-2
13.2. Understanding Exported Data 13-3
13.3. Exporting Data to a Database 13-5
13.3.1. Configuring an ODBC Data Source 13-8
13.3.2. Exporting DIS Data to a Database 13-9
13.3.3. Exporting HLA Data to a Database 13-10
13.3.4. The Export Log 13-10
13.4. Configuring a Data Source for Microsoft Access 13-11
13.5. Installing MySQL and Configuring a Data Source 13-14
13.5.1. Installing MySQL and the MySQL ODBC Driver 13-14
13.5.2. Creating a MySQL Database 13-15
13.5.3. Configuring a Data Source for MySQL 13-15
13.6. Exporting Data on Linux 13-18
13.7. Importing Logger Data into a Spreadsheet 13-19
13.7.1. Importing Data from MySQL to Microsoft Excel 13-19
13.7.2. Importing Data from Access to Excel 13-22

Chapter 14. Using Logger Commands

14.1. Using Logger Commands on the Command Line 14-2
14.2. Using Logger Commands in the Configuration File 14-3

14.3. Using Logger Commands in the Text Interface 14-3
14.3.1. Entering Commands 14-4
14.3.2. Displaying a List of Commands 14-5
14.4. Command Descriptions 14-5
14.4.1. Argument Types 14-6
14.4.2. System Domain Commands 14-7
14.4.3. Player Domain Commands 14-8
14.4.4. Recorder Domain Commands 14-10
14.4.5. Simulation Domain Commands 14-12
14.4.6. Editor Domain Commands 14-20
14.5. Command List 14-21

Chapter 15. Using the Logger Remote Control

15.1. The Logger Remote Control 15-2
15.2. Configuring Loggers to be Controlled Remotely 15-2
15.3. Starting the Logger Remote Control 15-3
15.4. Recording Exercises with the Logger Remote Control 15-3
15.5. Playing Exercises with the Logger Remote Control 15-5
15.6. Configuring the Logger Remote Control Connection 15-6
15.7. Specifying the Refresh Rate 15-6
15.8. Setting the Timebar Color 15-7
15.9. Exiting the Logger Remote Control and Connected Loggers .. 15-7
15.10. Logger Remote Control Command-line Options 15-8

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

Appendix B. FED File Naming Conventions

B.1. FED File Naming Convention B-2

MAK Products Glossary

Index

MAK Data Logger Quick Reference Card

Preface

The MAK 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.

Please consult *MAK 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 either the HLA or the DIS protocol.

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 Logger Files*, shows how to record network traffic and save the recordings.

Chapter 8, *Playing Logger Files*, shows how to play Logger files.

Chapter 9, *Annotating Files* explains how to create and edit annotations in Logger files.

Chapter 10, *Filtering Input and Output* explains how to use the Logger's filtering capabilities to limit the messages being sent or received.

Chapter 11, *Analyzing HLA Data* explains how to use the HLA Traffic Analyzer to understand the HLA data that the Logger is receiving or sending.

Chapter 12, *Editing Logger Files* explains how to edit Logger files, including cutting segments, inserting files, and offsetting packet timestamps.

Chapter 13, *Exporting Logger Data* explains how to export Logger files to other formats, such as text, and to MySQL or Microsoft Access databases.

Chapter 14, *Using Logger Commands*, explains how to execute Logger commands from the command line, from a console window, and from the configuration file.

Chapter 15, *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.

Appendix B, *FED File Naming Conventions*, maps old FED file names to current FED file names.

The *MAK Products Glossary* defines terms used in the manual.

The *MAK 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

Logger documentation is provided as manuals in PDF format, online help, and class documentation. Electronic versions of the documentation are in *logger/doc*. On Windows, you can access the documentation from the Logger folder on the Start menu. You can also find the most current documentation at https://www.mak.com/support/product-user-guides. The Logger documentation set is as follows:

- *MAK Data Logger First Experience Guide* introduces the Logger by walking you through some simple exercises.
- *MAK Data Logger Users Guide* describes how to install and use the Logger.
- API documentation and a class reference based on the header files is provided in HTML format.
- Online help. The online help, accessible from the Help menu, replicates most of the information in *MAK Data Logger Users Guide*.
- *MAK Products Interoperability Guide* describes general issues of HLA and DIS interoperability and specific issues for running MAK products together.
- *MAK Data Logger Release Notes* lists system requirements, release-specific requirements, new features and updates, bug fixes, and known problems.

MAK Products

The MAK Technologies product line includes the following:

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

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

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

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

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

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

  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MAK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- RadarFX. RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.

- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.
- MAK Legion. MAK Legion is a collection of applications and an SDK designed to support simulations with millions of simulation objects.

How to Contact Us

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

Telephone

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

E-mail

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

Internet

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

Post

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

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

Document Conventions

This manual uses the following typographic conventions:

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

Directory names are preceded with dot and slash characters that show their position with respect to the Logger home directory. For example, the directory *logger5.9/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

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

Third Party Licenses

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

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

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

The following libraries are covered by the ZLib license:

- Bullet Physics
- LibPNG
- LibSDL
- Minizip
- Zlib

The following libraries are covered by the MIT license:

- Expat
- FreeGLUT
- GDAL
- GeoTiff
- GLM
- HarfBuzz
- LibCURL
- LibSquish
- LibXML
- LUA
- LuaBind
- Proj
- SMHasher
- TCLAP
- Qt Visual Graph Editor

The following libraries are covered by the BSD license:

- FreeType
- GLEW
- JPEG
- LibTiff
- Protobuf
- PCRE
- Ogg Vorbis

The following libraries are covered by the commercial licenses:

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

The following libraries are covered by custom licenses:

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

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLLite is in the public domain.

SWIG is covered by the GPL license.

Boost License

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

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

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

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

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

libXML and libICONV

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

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

QScintilla and SciTE

License for Scintilla and SciTE. Copyright 1998-2002 by Neil Hodgson <neilh@scintilla.org>. All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS

SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

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 1-3
- Streaming Video 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 exercises[1] 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 2-10 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 10, *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 9, *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 12, *Editing Logger Files*.

1.1.5. Streaming Video

The Logger can record or play a video stream file while synchronously recording or playing a recording of the related simulation exercise traffic.

The video stream can be captured from the network and saved to a separate video file. The video stream can come from VR-Vantage, a webcam, an external live camera, or from any source generating a stream over the network. For details, please see "Recording Exercises," on page 7-2.

The video file can be played back to an embedded video window, much like using VLC or another media player. The video file can also be streamed back onto the network. If you have an application that was reading video stream data during the exercise, it will now be able to read the same data when the Logger is playing back the exercise. The video playback will be synchronized to the Logger data stream. So if you are playing back your exercise at 2x speed, the video playback speeds up as well. Similarly, if you jump to a specific point in time, both the video and the simulation data jump to that location. For details, please see "Enabling Video Recording," on page 7-7 and "Viewing Video," on page 8-3.

1.1.6. 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 13, *Exporting Logger Data*.

Developers can add additional export formats using the Logger API.

1.1.7. 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.

1.1.8. 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.

1.1.9. Sample Logger Files

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

- *bombMission2016*, *bombMission2018*. (DIS, HLA 1.3, HLAe) The mission is to drop a bomb on a high priority target on the island of Oahu, Hawaii. To accomplish the mission, several F-16 decoys are used to distract enemy forces while a B-2 drops a high altitude JDAM on the primary target.
- *BrooklynBomb-2019*. (DIS, HLAe) In this scenario, a bomb is detonated in a church parking lot during a protest. It includes an ISR sensor view. Terrain to load with: Brooklyn.
- *FollowSuspect-2019*. (DIS, HLA 1.3, HLAe) This scenario follows human suspects through a section of Honolulu, Hawaii. The suspect moves through the scenario using various vehicles and retrieves a weapon while an overhead UAV tracks the events.
- *hawaiiNaval2016*. (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.
- *hawaiiTour2019*. (DIS, HLA 1.3) The Hawaii Tour scenario showcases a helicopter flying around the island of Oahu.
- *ProceduralEarth*. (DIS) Several different recordings demonstrate MAK Technologies's procedural terrains. Terrain to load with: ProceduralEarth.
- *Raid2020*, *Raid-SFX-ViewControl-DIS*. (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 version is designed for use with SensorFX.
- *rhoneValleyTour-2019-DIS*. The Rhone Valley Tour scenario showcases two small aircraft flying around the valley.
- *VehicleLights2015-DIS*. Demonstrates lighting effects and lights cast by a HMMWV on the Hawaii terrain.
- *VR-Engage_FirstExperienceDIS*. This is a webcam recording of some engineers going through the First Experience scenario for VR-Engage 1.3.
- *VR-TheWorld_FirstExperience*. (DIS, HLA 1.3) RPR FOM 1.0. This recording is for use with *VR-TheWorld First Experience Guide*.

Most of these files contain View Control messages, which cause MAK'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.

1.1.10. Sample Programs

The Logger includes some example programs:

- Simple examples of using the Logger API — playFile, readFile, recordFile, writeFile, and exportFileToDatabase. Compiled versions of these examples are provided in the *./bin64* 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. Look for these in the *./examples* directory.
- An example Logger application — *loggerApp*. You can use this example as a starting point for changing and recompiling the Logger application. It is also in the *./examples* directory.
- A remote control application — *lgrControl*. Compiled versions of the example are provided in the *./bin64* directory.
- A DDM example — *sampleDdm* in the *./examples* directory shows how to manipulate DDM regions using packet data.
- *lgrModSqlExporter* — An example that extends the database export to include the DIS Data PDU. This is also in the *./examples* directory.
- *matrexTspiFomMapper* — A VR-Link FOM Mapper for the entity attributes of the MATREX FOM for Time and Space Position Information. This is also in the *./examples* directory.

For more information about these and other 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 *MAK 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 MAK 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 MAK Data Logger 2-2
Installing MAK Data Logger on Windows 2-2
Installing Logger on a Linux System 2-3
The Logger Directory Structure 2-4
Installing and Setting Up the MAK License Manager 2-5
Specifying the License Server 2-6
Installing an RTI 2-8
Installing the MAK RTI 2-8
Localizing the Graphical User Interface 2-10
Translating Other Interface Files 2-13
Applying the Language Files 2-14

2.1. Installing MAK Data Logger

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

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

2.1.1. Installing MAK Data Logger on Windows

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

Windows versions of 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 Logger

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

2.1.2. Installing Logger on a Linux System

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

To install Logger on Linux:

1. Create the directory in which you want to install 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.

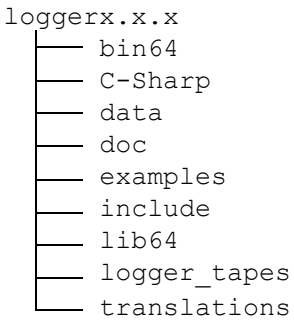


Figure 2-1. Logger directory structure

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

Table 2-1: Logger directory structure

Directory	Contents
bin64	Executables and DLLs.
C-Sharp	C# Remote Control API and examples.
data	Subdirectories contain default configuration files and images used in the GUI.
doc	Documentation.
examples	Examples of the Logger API and the Logger File API.
include and *lib64*	Header files and libraries for the Logger API and Logger File API.
logger_tapes	Sample Logger files.
translations	Translation files for localizing the Logger.

2.3. Installing and Setting Up the MAK License Manager

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

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

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

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 MAK 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.

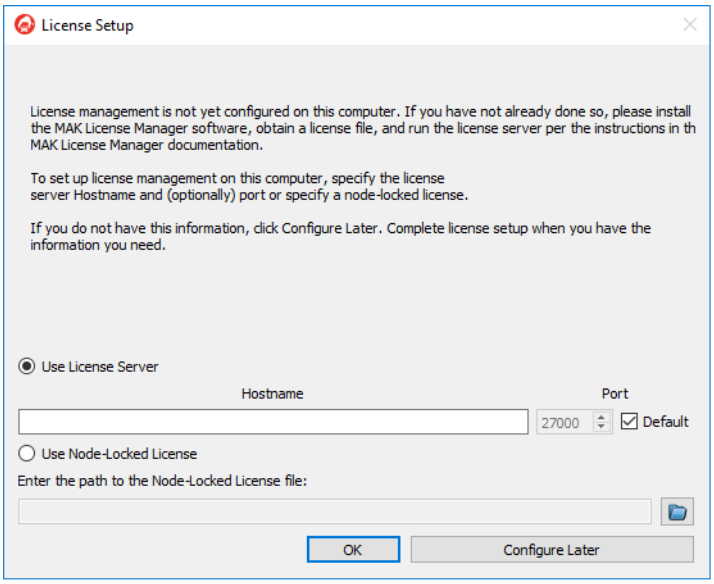


Figure 2-2. License Setup dialog box

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 MAK applications until you set up license management.

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

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

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

The MAKLMGRD_LICENSE_FILE Environment Variable

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

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

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

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

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

Linux

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

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

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

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

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

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

2.4. 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 MAK products, please see the release notes for your MAK application.

2.4.1. Installing the MAK RTI

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

Configuring Your System to Use the MAK RTI

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

- Linux — LD_LIBRARY_PATH
- Windows — PATH

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

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

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

Running Applications with the MAK RTI

To run a MAK application with the MAK RTI:

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

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

2.5. Localizing the Graphical User Interface

You can localize your MAK 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 MAK 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).

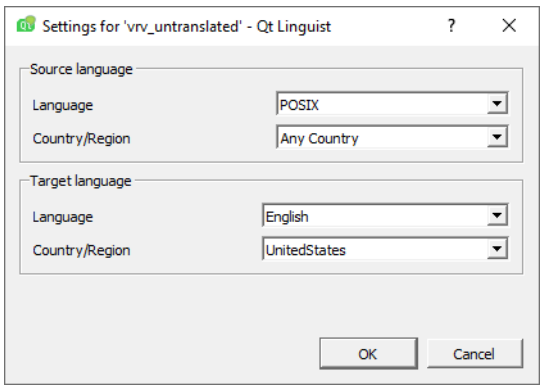


Figure 2-3. Linguist settings dialog box

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).

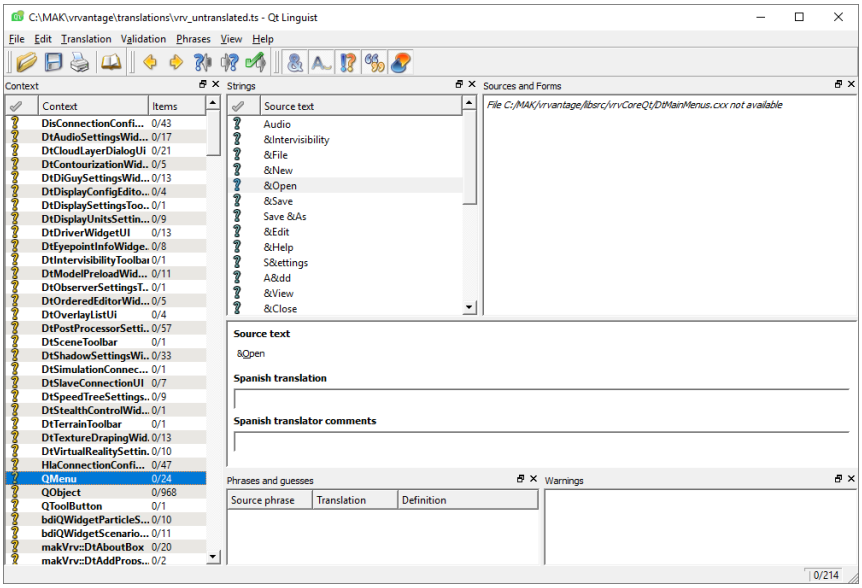


Figure 2-4. Linguist window

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.

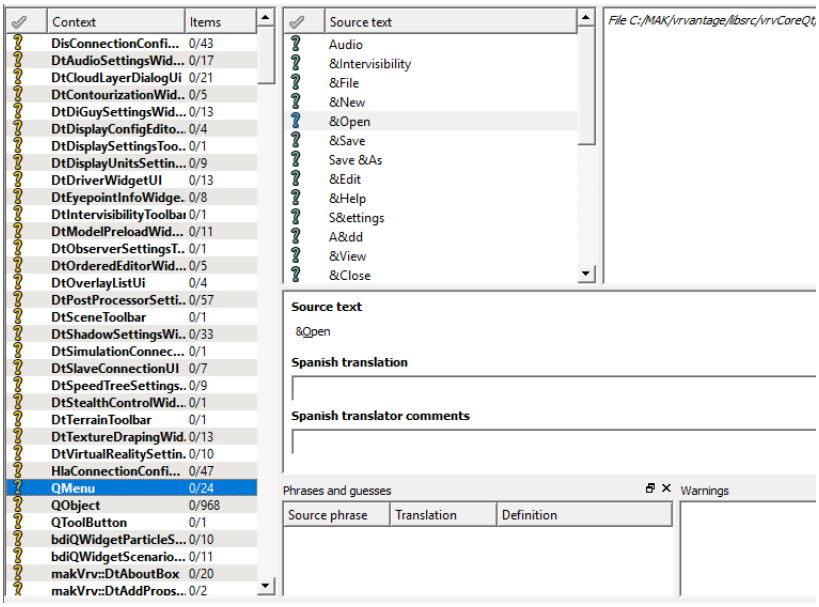


Figure 2-5. Text selected for translation

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.

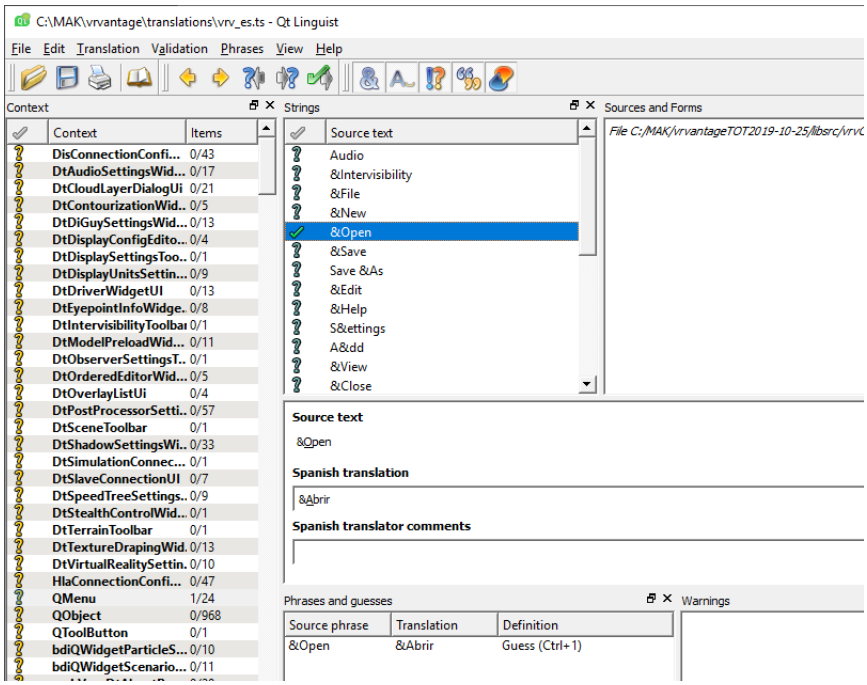


Figure 2-6. Completed translation for Open

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 *logger_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 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 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.

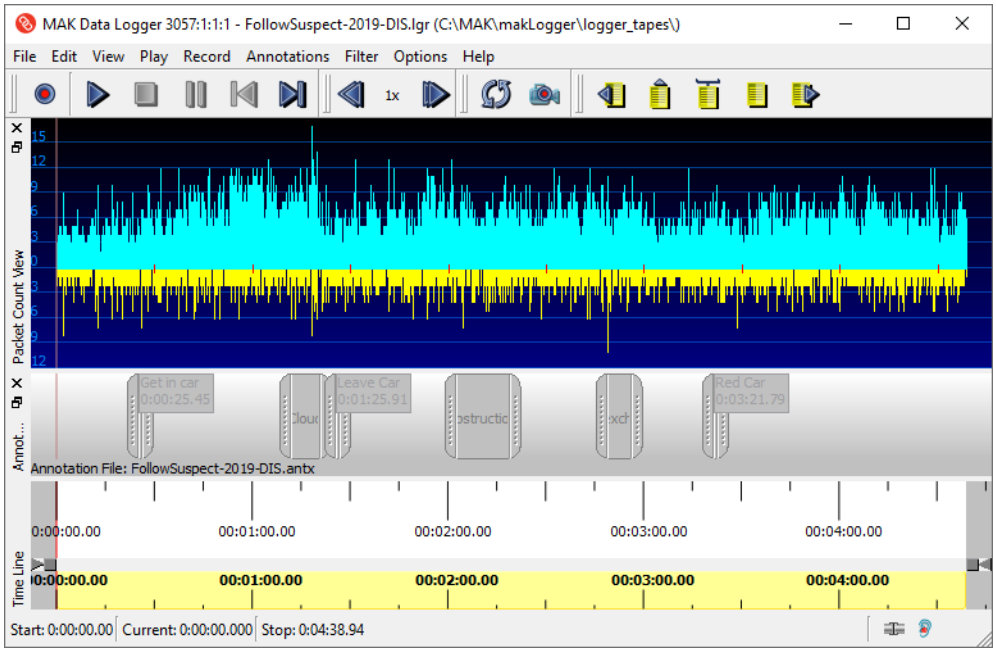


Figure 3-1. The Logger window

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 the appropriate *lgrConfig.xml* file, 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*, *lgrConfigHLA1516.xml*, and *lgrConfigHLA1516e.xml* in the *bin64* directory). For simplicity, this manual refers to these files 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 timestamp 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 8-11.

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 MAK 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 8-6.

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 4-24.

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 4-25.

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 *MAK 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 *MAK 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 4-10
Ignoring Advisory Messages 4-11
Specifying a FOM Mapper 4-11
Configuring Time Management 4-14
Specifying the Logger Subscription List 4-15
Subscribing Passively to Objects and Interactions 4-17
Enabling Request Attribute Updates 4-18
Sending Only Updated Values 4-18
Configuring the Logger for DIS 4-19
Specifying a UDP Port 4-20
Setting the Exercise ID 4-20
Setting the Site ID 4-20
Setting the Application Number 4-21
Using Asynchronous I/O 4-21
Specifying a Destination Address for Outgoing and Incoming Data 4-21
Setting the Subnet Mask 4-22

Setting the Buffer Size for Sending and Receiving PDUs 4-22
Setting the Multicast Time to Live Value 4-22
Setting the Network Interface Address 4-23
Specifying the Heartbeat 4-24
Specifying the Timeout 4-25
Specifying the DIS Version 4-25
Subscribing to Multicast Groups 4-26

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).

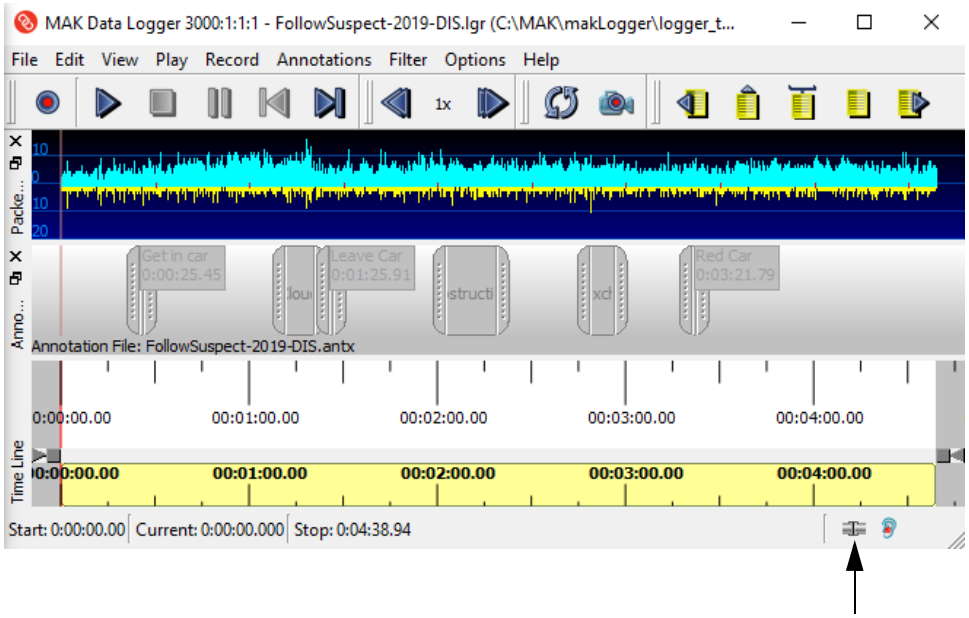


Figure 4-1. Connection status

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 4-19.

By default, when you change connection properties, they are saved to the *lgrConfig.xml* configuration file. You can enable and disable this setting in 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)).

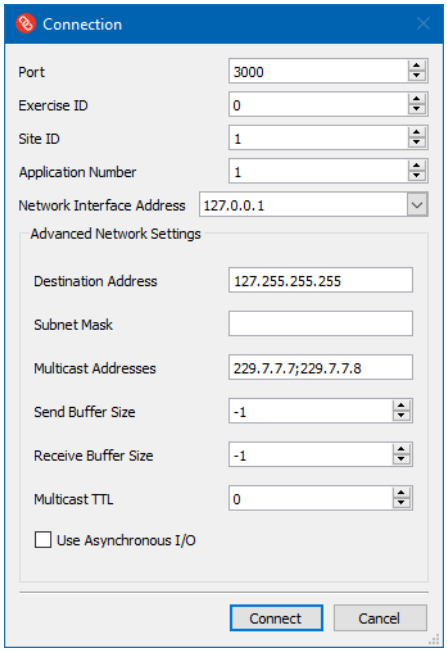


Figure 4-2. Connection dialog box for DIS

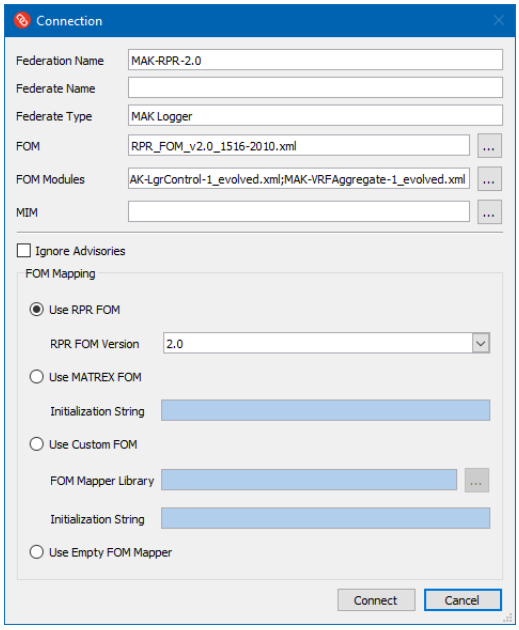


Figure 4-3. Connection dialog box for HLA Evolved

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:

```
SetConnectionParametersHLA federateName federationName
   fedFile ignoreAdvisories fomVersion fomRevision
   useMatrexFomMapper matrexFomMapperData fomMapper
   fomMapperData useEmptyFomMapper noRtiCompilerCheck
```

For example:

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

The syntax for `SetConnectionParameters` for HLA Evolved is:

```
SetConnectionParametersHLAe federateName federateType
   federationName fedFile "fomModules" MIM ignoreAdvisories
   fomVersion fomRevision useMatrexFomMapper
   matrexFomMapperData fomMapper fomMapperData
   useEmptyFomMapper noRtiCompilerCheck
```

For example:

```
SetConnectionParametersHLAe Logger "MAK Logger"
   MAK-RPR2.0 MAKRPR2-1-1.xml "VR-Link_evolved_module.xml
   extendF18Module.xml" HLAstandardMIM.xml False 1.0 1
   False "" myFomMapper "myInitParameter1 myInitParameter2
   …" False False
```

The *`fomMapperData`* and *`matrexFomMapperData`* arguments can be lists. If so, you must wrap each list 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:

```
SetConnectionParametersHLA myLogger MAK-RPR2.0
   MAK-RPR2-1-1.fed False 2.0 1 False "" myFomMapper
   "myInitParameter1 myInitParameter2" False False
```

For details about commands and command syntax, please see Chapter 14, *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	`logger`*`version`* `-N myLogger`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The federateName 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	`logger`*`version`* `--federateType "my custom Logger"`
Text interface	Use the `SetConnectionParameters` command.
lgrConfigHLA1516e.xml	The federateType 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	`logger`*`version`* `-x MAK-RPR-2.0`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The execName 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	`logger`*`version`* `-F MAK-RPR-2.0.fed`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The fedFileName 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. MAK 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	`logger`*`version`* `--fomModules "module1.xml module2.xml ... modulen.xml"`
Text interface	Use the `SetConnectionParameters` command.
lgrConfigHLA1516e.xml	The `fomModules` 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. MAK products come with the standard MIM. When you specify MIM file, the Logger looks for it in its *bin64* directory.

➤ 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	`logger`*`version`* `--MIM HLAstandardMIM.xml`
Text interface	Use the `SetConnectionParameters` command.
lgrConfigHLA1516e.xml	The `MIM` 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 *MAK 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	`logger`*`version`* `--rprFomVersion 2.0017 -F VR-Link20017-1.fed`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The `rprFomVersion` 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	`logger`*`version`* `--rprFomVersion 2.0017 --rprFomRevision 1 -F VR-Link20017-1.fed`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The rprFomRevision 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	`logger`*`version`* `--ignoreAdvisories 1`
Text interface	Use the `SetConnectionParameters` command, or use `SetIgnoreAdvisories true`
lgrConfig.xml	The ignoreAdvisories variable is in the default configuration file. You can also add `SetIgnoreAdvisories` 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	`logger`*`version`* `-f myFomMapper.dll`
	or
	`logger`*`version`* `--useEmptyFomMapper 1`

Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The fomMapperLib and useEmptyFomMapper 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	`logger`*`version`* `-f myFomMapper.dll --fomMapperData ABC` where ABC is passed to the FOM Mapper creator function in the shared library as an initialization parameter.
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The `fomMapperData` 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).

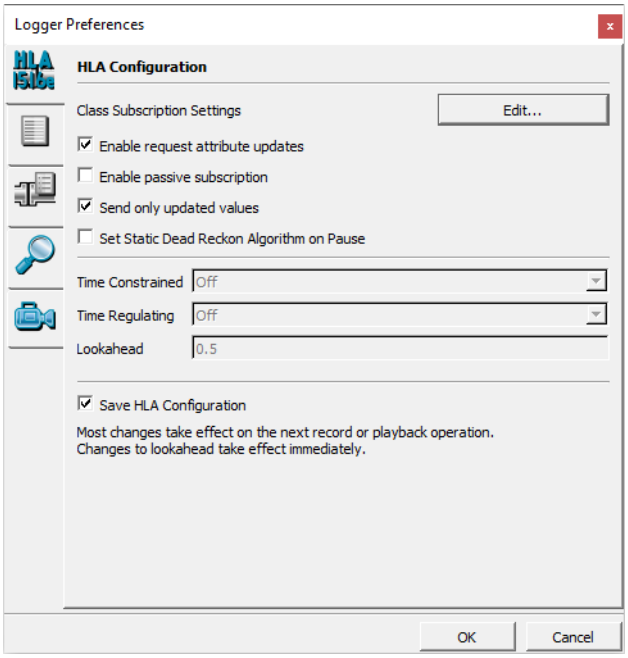


Figure 4-4. Logger Preferences dialog box, HLA Configuration page

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	`logger`*`version`* `-r "SetTimeManagement true false false true 0.5"`
Text interface	`SetTimeManagement true false false true 0.5`
lgrConfig.xml	Add `SetTimeManagement` 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.

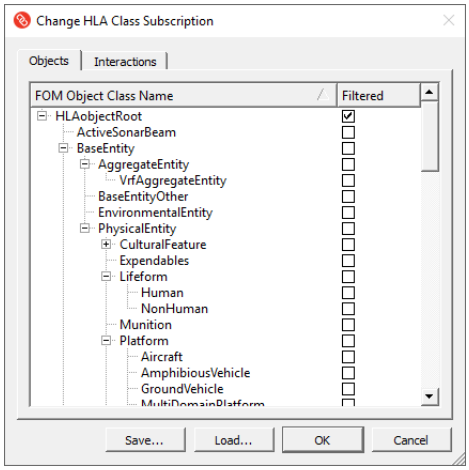


Figure 4-5. Change HLA Class Subscription dialog box

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	`logger`*`version`* `-r "SetSubscriptionLists \"ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft ObjectRoot.BaseEntity.PhysicalEntity.Radio InteractionRoot.WeaponFire\""`
Text interface	`SetSubscriptionLists "ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft ObjectRoot.BaseEntity.PhysicalEntity.Radio InteractionRoot.WeaponFire"`
lgrConfig.xml	Add `SetSubscriptionLists` 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 5-27. ! The `SetSubscriptionLists` command must come after the `Connect` 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	`logger`*`version`* `-r "SaveSubscriptionListFile mySubList.subx"`
Text interface	`SaveSubscriptionListFile mySubList.subx`

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	`logger`*`version`* `-r "LoadSubscriptionListFile \"../config/mySubList.subx\""`
Text interface	`LoadSubscriptionListFile ../config/mySubList.subx`
lgrConfig.xml	Add `LoadSubscriptionListFile` 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	`logger`*`version`* `-r "SetPassiveSubscription true"`

Table 4-16: Specifying passive subscription

Interface	Command
Text interface	`SetPassiveSubscription true`
lgrConfig.xml	Add `SetPassiveSubscription` 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	`logger`*`version`* `-r "EnableRequestUpdates true"`
Text interface	`EnableRequestUpdates true`
lgrConfig.xml	Add `EnableRequestUpdates` 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	`logger`*`version`* `-r "SetSendOnlyUpdatedValues true"`
Text interface	`SetSendOnlyUpdatedValues true`
lgrConfig.xml	Add `SetSendOnlyUpdatedValues` 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:

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

For example:

```
loggerDIS.exe -r "SetConnectionParametersDIS 3000 1 1 2
   True -1 -1 -1 127.255.255.255 255.0.0.0 127.0.0.1
   \"229.7.7.8\""
```

When you use the `SetConnectionParametersDIS` 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:

```
loggerDIS.exe -r "SetConnectionParametersDIS 5000 1 1 2
   True -1 -1 -1 127.255.255.255 255.0.0.0 127.0.0.1
   \"229.7.7.8\""
```

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 14, *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	`loggerDIS -P 2345`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The disPort 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	`loggerDIS -e 3`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The exerciseId 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	`loggerDIS -s 3`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The siteId 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	`loggerDIS -a 2`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The appNumber 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 `SetConnectionParameters` command.
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The useAsyncIO 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	`loggerDIS -A 205.205.205.205`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The destAddrString 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 `SetConnectionParameters` command.
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The subnetMask 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	`loggerDIS --receiveBufferSize 64000 --sendBufferSize 64000`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The sendBufferSize and recvBufferSize 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	`loggerDIS --mcastTtl 3`
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The mcastTtl 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	`loggerDIS --deviceAddrString` *string*
Text interface	Use the `SetConnectionParameters` command.
lgrConfig.xml	The deviceAddrString 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).

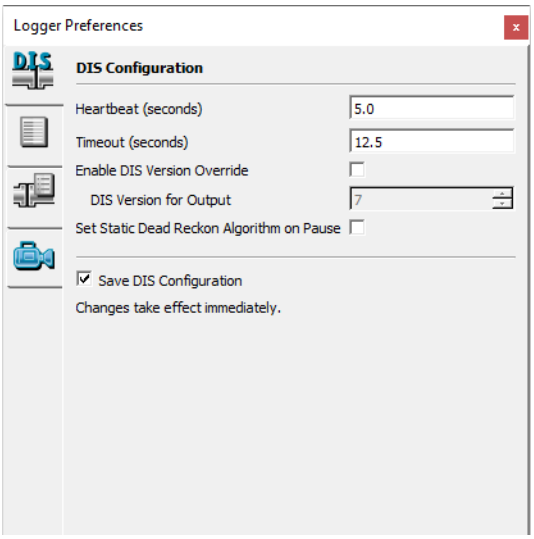


Figure 4-6. Logger Preferences dialog box, DIS Configuration page

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	`loggerDIS -r "SetHeartbeatRate 5.0"`
Text interface	`SetHeartbeatRate 5.0`
lgrConfig.xml	Add `SetHeartbeatRate` 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	`loggerDIS -r "SetTimeoutRate 12.0"`
Text interface	`SetTimeoutRate 12.0`
lgrConfig.xml	Add `SetTimeoutRate` 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	`loggerDIS -S "225.0.0.118 225.0.0.202"`
Text interface	`SetConnectionParameters` *`port exerciseId siteId appNumber useAsynchIO sendBufferSize recvBufferSize multicastTtl destAddr deviceAddr "mcastAddresses"`*
lgrConfig.xml	The `mcastAddresses` 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 5-2
Configuring Indexing Using the Command Interfaces 5-4
Configuring Faster Playback 5-4
The SetEntityTracking Command 5-5
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 5-18
Setting the Notification Level 5-19
Specifying the Log File 5-19
Configuring Streaming Video 5-20
Video Record Settings 5-20
Video Playback Settings 5-20
Video Advanced Settings (Configuring GStreamer) 5-21
Editing lgrConfig.xml 5-23
Adding a Command to lgrConfig.xml 5-26
Summary of Command-line Options for the Logger 5-28

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 5-28 or the Quick Reference Card. For descriptions of Logger commands and how to use them, please see Chapter 14, *Using Logger Commands* or the Quick Reference Card. For details about the configuration file, please see "Editing lgrConfig.xml," on page 5-23.

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. (As a result, a single aggregated update is sent for each active entity.) 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.)

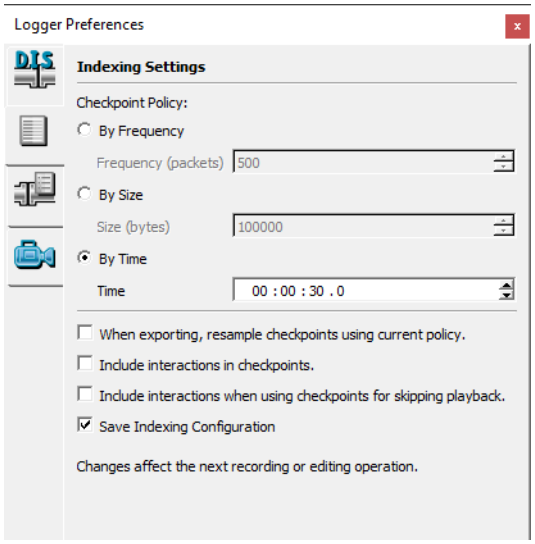


Figure 5-1. Logger Preferences dialog box, Indexing Settings page

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	`logger`*`version`* `-r "SetCheckpointParameters 0 0 500 0"` For details about the parameter values, please see "Recorder Domain Commands," on page 14-10.
Text interface	`SetCheckpointParameters 0 0 500 0`
lgrConfig.xml	Add `SetCheckpointParameters` 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. 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 and a single aggregated update is sent over the network for each active entity. 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 closet 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.)

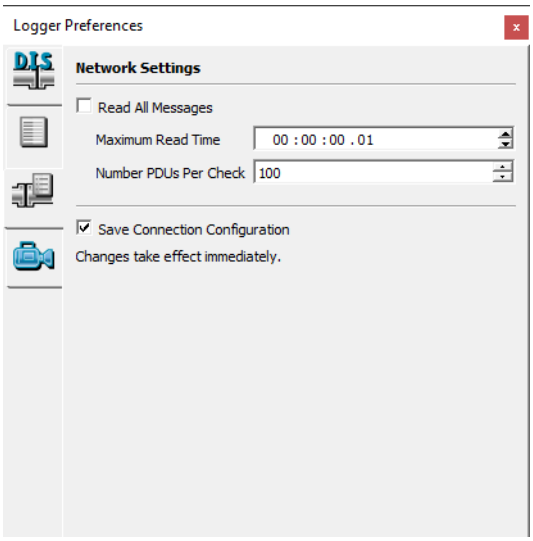


Figure 5-2. Logger Preferences dialog box, Network Settings page

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. -->
  <1-- 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	`logger`*`version`* `-m 2 -n 2`
	or
	`logger`*`version`* `-r "SetNotifyLevel 2 2"`

Text interface	`SetNotifyLevel 2 2`
lgrConfig.xml	The `SetNotifyLevel` 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	`logger`*`version`* `-l mylogfile.log`
Text interface	None.
lgrConfig.xml	The logFileName variable is in the default configuration file.

5.8. Configuring Streaming Video

You can change the default settings for recording and playing video and you can configure GStreamer.

5.8.1. Video Record Settings

You can configure whether or not video recording is enabled at startup and you can specify the port to use.

To configure video recording:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the Video Settings page (Figure 5-3).

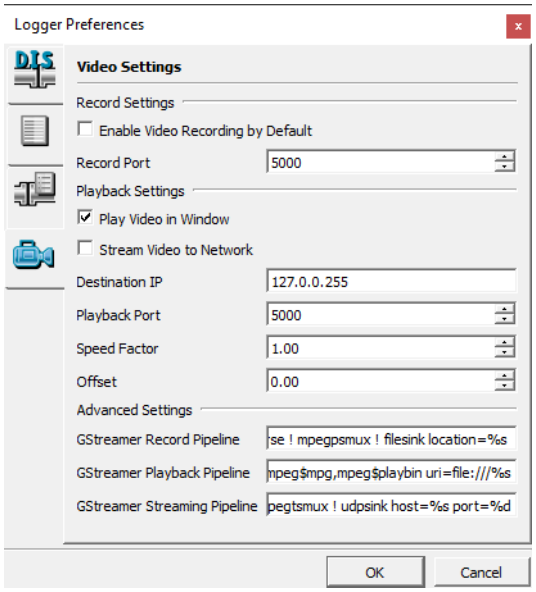


Figure 5-3. Video Settings page

3. To enable video recording at startup, select the Record Videos check box. To disable it, clear the check box.
4. To set the port used for recording, change the value in the Record Port box.
5. Click OK.

5.8.2. Video Playback Settings

You can enable or disable video playback at startup and whether or not to stream to the network. You can also set the following parameters:

- Port. The port to which to stream video over the network.
- Destination IP address. The destination IP address of the stream. This value should be an appropriate device IP, multicast IP, or broadcast IP.

Only one consumer can read a video stream sent to a device IP address.

- Speed Factor. If the video and data recordings are not synchronized, you can adjust the speed of video playback relative to the data playback to try to synchronize them. This factor is applied in addition to any changes to the data playback rate. Default: 1.
- Offset. If the video file starts before or after the data recording, you can specify an offset for the video so that it starts at the correct time relative to the data. Default: 0.

The Speed Factor and Offset affect all videos. Therefore, if you need to set them for a particular video, you should reset them to the default values before running other videos.

To configure video playback:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the Video Settings page (Figure 5-3).
3. To enable or disable video playback in the Logger window at startup, select or clear the Play Video in Window check box.
4. To enable or disable playing video to the network, select or clear the Stream Video to Network check box.
5. To specify an destination IP address, type an IP address in the Destination IP box.
6. To specify the port for playback, specify a value in the Playback Port box.
7. To set a speed factor to synchronize video to data, specify a value in the Speed Factor box.
8. To set an offset to synchronize a video and data recording, specify a value in the Offset box.
9. Click OK.

5.8.3. Video Advanced Settings (Configuring GStreamer)

Video streaming in the MAK Logger uses the open source library GStreamer. GStreamer provides many options that allow it to record different video files and to resize, transform, complex, and multiplex the video. The default pipelines support video files containing a single video stream using the MPEG format.

You can change the configuration used by the Logger by editing the record and playback pipelines. Do not edit them unless you are familiar with building GStreamer configuration pipelines. For information about GStreamer and how to build a configuration pipeline, please go to http://gstreamer.freedesktop.org.

If you modify a pipeline and want to return it to its original text, you can copy the appropriate text string from one of the protocol-specific configuration files in *./data/config*. Copy the text either to the similar configuration file in *./bin64* or paste the text string into the correct pipeline box on the Logger Preferences dialog box, Video Settings page.

The lines to look for in the configuration file are as follows:

```
<!-- %s in record pipeline is replaced with logger tape file with ".lgr"
  extension stripped %d is replaced with the port -->
<var name="mediaRecordType" type="string" value="mpeg$mpg$udpsrc port=%d
  ! tsdemux ! mpegvideoparse ! mpegtsmux ! filesink location=%s"/>

<!-- %s in playback pipeline is replaced with media file name -->
<var name="mediaPlayInWindowType" type="string"
  value="mpeg$mpg,mpeg$playbin uri=file:///%s"/>

<!-- %s in record pipeline is replaced with output host
%d is replaced with the output port -->
<var name="mediaPlayToStreamType" type="string"
  value="mpeg$mpg,mpeg$video/x-raw,framerate=25/1 ! videoconvert !
  avenc_mpeg2video ! mpegtsmux ! udpsink host=%s port=%d"/>
```

The configuration file also contains an H264 configuration, which is commented out. The H264 configuration provides better video quality, but MPEG provides better support for jumping to different locations in the video.

5.9. 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 *./bin64* 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 14-5.)

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 occuring.

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 14-5 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 14, *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 14-5.
- *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 14-5.
- *value* is the value you want to assign to the variable.

For example:

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

5.9.1. 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 14-5. 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.

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>
```

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 14-2 and "Using Logger Commands in the Text Interface," on page 14-3.

5.10. 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
`--allowAllRemoteControl` *mode*	Specifies the response to remote control PDUs. 1 = Allow 2 = Do not allow
`{-c | --showConsole}` *mode*	Enables (1) or disables (0) display of a console window.
`--configFile` *filename*	Specifies the configuration file to load.
`--defaultTapeDir` *directory*	Specifies the default directory for Logger files.
filename	Specifies a Logger file to load. This option uses an implicit `-r OpenFile`. If a filename starts with a hyphen (-), you must use the `-r OpenFile` syntax.
`--filterEntityTypes` *entity_type*	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.
`{-- | --ignore_rest}`	Ignore all arguments after this one.
`{-h | --help}`	Displays a brief summary of Logger command-line arguments.
`{-l | --logFileName}` *filename*	Specifies a file to send notification messages to. Default: *logger.log*.
`--loggerTickRate` *rate*	The interval for the Logger tick rate, in milliseconds. For more information, please see "The SetPlayback-ReadTimeLimit Command," on page 5-7
`{-m | --messageLevel}` *level*	Specifies the notification level for Logger messages for the log file. Range: 0-4.
`{-n | --notifyLevel}` *level*	Specifies the notification level for VR-Link messages for the log file. Range: 0-4.
`--remoteControlId` *ID*	Specifies the Logger's remote control ID. By default the value is assigned the Logger's site ID. This value is used if `SetRemoteControlSettings allowAllRemoteControl` is disabled.
`{-r | --runCommand}` *command*	Specifies a Logger command to run. You can specify `-r` multiple times. For more information, please see "Using Logger Commands on the Command Line," on page 14-2.
`{-v | --version}`	Displays version information and exits.

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

Option	Description
HLA Only	
`--disableTMTesting` *mode*	Disable time management for testing.
`--enableTrafficAnalyzer` *mode*	Enable (1) or disable (0) the HLA Traffic Analyzer.
`{-f | --fomMapperLib}` *libname*	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 4-11.
`--federateType` *type*	Specifies the federate type for the Logger federate.
`--filterObjectNames` *names*	Filter the listed object names. Accepted multiple times.
`--fomMapperData` *data*	Specifies FOM Mapper initialization data, if required.
`--fomModules` *module*	Specifies a FOM module for the exercise. (accepted multiple times, HLA Evolved only)
`{-F | --fedFileName}` *fedfile*	Specifies the FED file for the exercise.
`--ignoreAdvisories` *mode*	Specifies whether (1) or not (0) the Logger ignores advisory messages.
`--maxDrainInputTime` *time*	Specifies the maximum time to drain input.
`--MIM` *MIM_file*	Specifies the MIM (MOM and Initialization Module) for the exercise. (HLA Evolved only)
`{-N | --federateName}` *federate_name*	Specifies the federate name for the Logger federate. (HLA Evolved only)
`--rprFomVersion` *version*	Specifies the version of the RPR FOM to use.
`--rprFomRevision` *revision*	Specifies the FOM revision to use.
`--timeConstrainingRecord-ing` *mode*	Use time constrained for recording.
`--timeRegulatingPlayback` *mode*	Use time regulating for playback
`--useEmptyFomMapper` *mode*	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 4-11.
`{-x | --execName}` *ExName*	Specifies the federation execution name. Default: VR-Link.
DIS Only	
`{-A | --destAddrString}` *address*	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 4-21.

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

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

Connecting to the Simulation Network 6-6
- Disconnecting from the Simulation Network 6-6

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 6-10

Viewing and Editing Logger File Information 6-10
- Displaying Information about a Logger File 6-11
- Editing the Logger File Description 6-11

6.1. Starting the Logger

The Logger has separate executables for each protocol supported: *loggerDIS*, *loggerHLA13*, *loggerHLA1516*, and *loggerHLA1516e* (please see Table 2-1 for a description of the directory structure). In sample command lines, we use the generic form logger*version*. 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 5-28, lists command-line options for the Logger.

6.1.1. Starting the Logger on Linux

To start the Logger:

1. Open a Linux shell.
2. Change to the *./bin64* directory.
3. Enter:

   `logger`*`version`* `[`*`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 Windows 10 Start Menu, choose **MAK Technologies** → **Logger** *protocol*, where *protocol* is DIS, HLA 1.3, or HLA 1516 Evolved.
- In a command window, change to the *./bin64* directory, and type:

  `logger`*`version`* `[`*`filename options`*`]`
- Create a shortcut on the Windows desktop and specify the appropriate command line in the Properties dialog box.
- In the File 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 14-3. 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 packet counts and annotations, is displayed between the toolbar controls and the timeline. You can enable and disable the display of plug-in data.

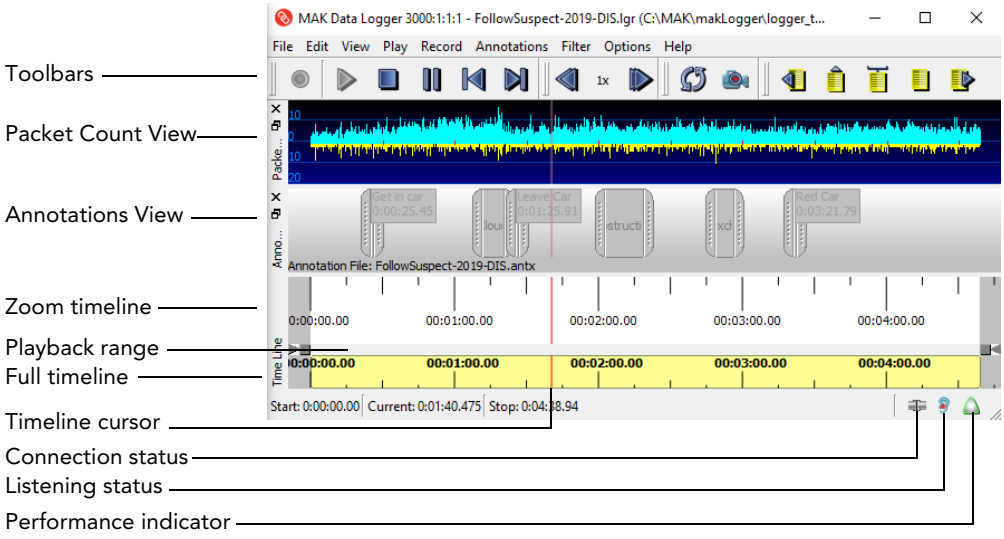


Figure 6-1. Logger window with a Logger file loaded

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).
- Enable video recording.
- 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.

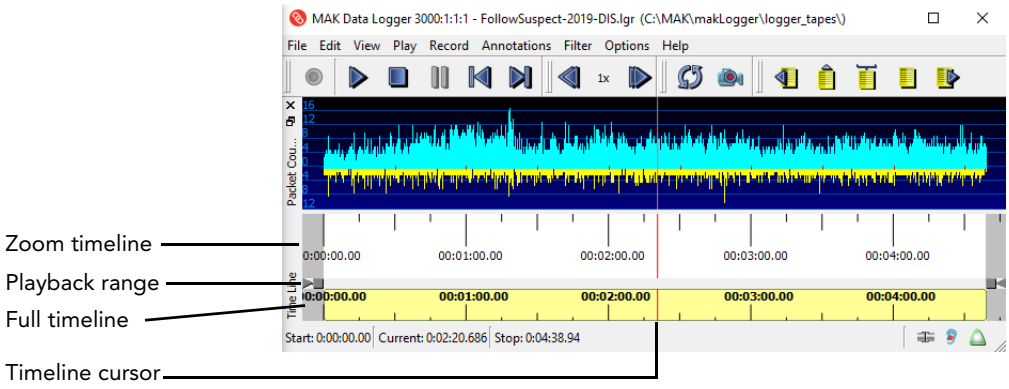


Figure 6-2. Timeline

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 8-3, "Zooming the Timeline," on page 8-9, and "Moving the Timepoint to a Specific Time," on page 8-11.

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).

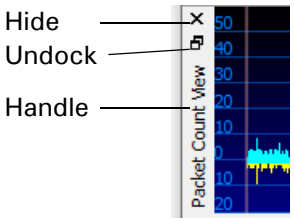


Figure 6-3. Plug-in window controls

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 anyplace 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.

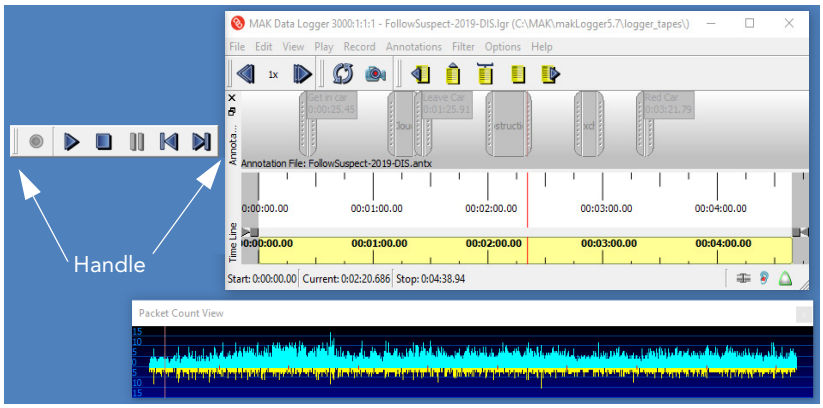


Figure 6-4. Undocked GUI components

- 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.

6.2.4. Saving the Window Layout

When you close the Logger, it automatically saves the window size and location to *logger-gui,conf* in your AppData directory on Windows and home directory on Linux (for example, *C:\Users\user_name\AppData\Roaming\MAK\Logger\5.6\logger-gui.conf* (Windows) or *~/.mak/Logger/5.6/logger-gui.conf* (Linux)). The name of the configuration file is specified in the configuration file for the version of the Logger you are running, for example, *./bin64/lgrConfigDIS.xml*.

You can change the name of the GUI configuration file by editing the following line in the Logger configuration file:

```
<var name="defaultGuiConfigFilename" type="string"
  value="logger-gui.conf"/>
```

`SetGuiConfigFilename` command.If you want to run multiple Loggers at the same time and automatically place them in different locations, you can specify different locations for the GUI configuration file either by using different Logger configuration files or the command line.

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 8-12.)

By default, if a recording has a video file associated with it, and if the video file has the same name as the recording (except for the extension), the Logger plays the video file in the Logger window when you play the recording. When you open a Logger recording that has an associated video file, you can also specify the name of the file and specify that it be streamed to the network.

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 ( ).

   If there is a video file that has the same name as the Logger file, it is listed in the file box in the Video Playback section of the Open File dialog box.

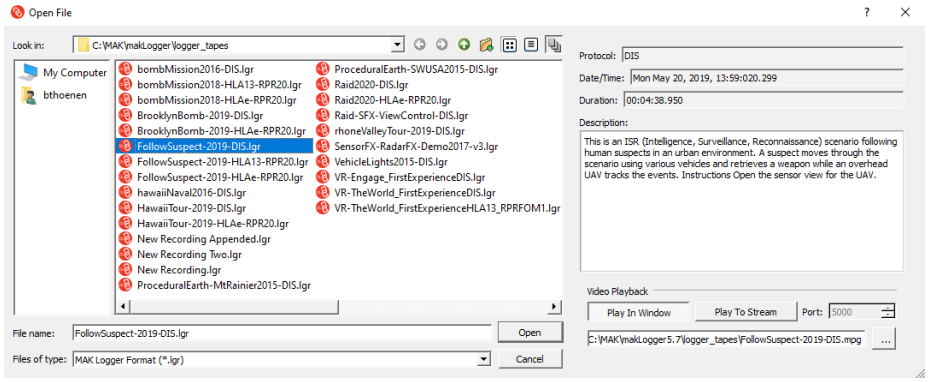


Figure 6-5. Open File dialog box

3. Configure video playback options as follows:
   a. If video playback is enabled, but you do not want to play the video in the Logger window, click the Play In Window button to disable playback.
   b. If video playback is disabled and you want to play a video recording in the Logger, click the Play In Window button to enable playback.
   c. If there is a video file associated with this Logger recording and it does not have the same name as the Logger file, specify it in the file box in the Video Playback section of the dialog box.
   d. If you want to stream the video to the network, click the Play To Stream button.
   e. Optionally, if Play To Stream is enabled, change the port. (You can specify the default port and IP address on the Logger Preferences dialog box, Video Settings page.)

The Play In Window and Play To Stream options are independent of each other.

4. Click Open. If there is an annotations file with the same name as the Logger file, it opens automatically. If Play On Load is enabled, the Logger starts playing the file immediately. If there is a video file for this recording and the Video Stream view is not open, it opens automatically.

- 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.

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_filename*Repaired.*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.

When idle listening is enabled before recording starts, an "initial state" packet is inserted at the beginning of the recording. It contains the current known state of any entities. When the recording is played back, the initial state produces appropriately dead reckoned states to represent the entities from the start time of the recording.

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).

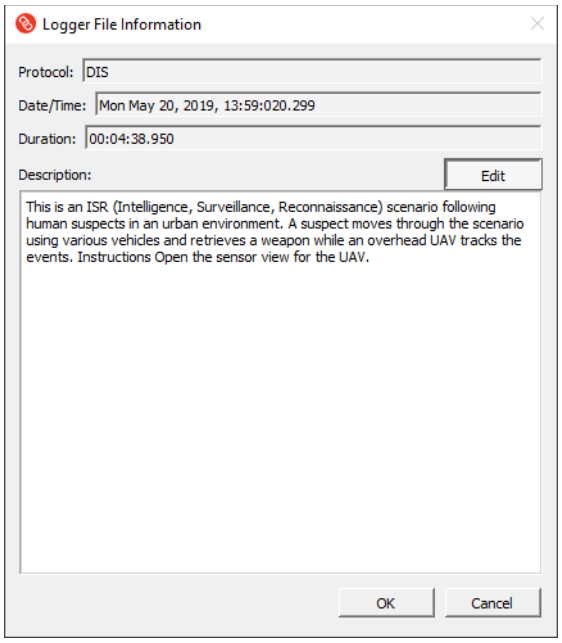


Figure 6-6. Logger File Information dialog box

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).

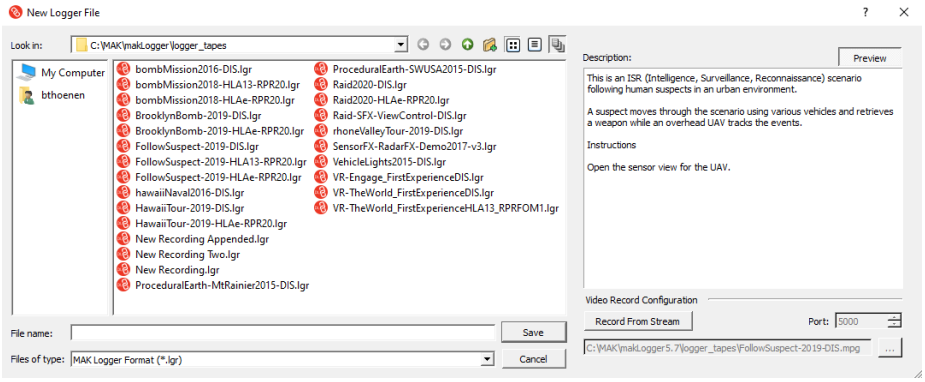


Figure 6-7. New Logger File dialog box

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).

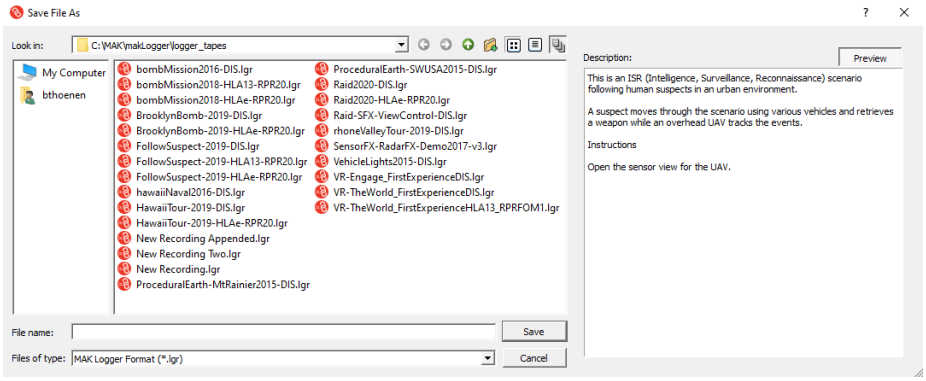


Figure 6-8. Save File As dialog box

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 Logger Files

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

Recording Exercises 7-2
- Recording Immediately (to a Temporary File) 7-2
- Recording to a Specified File 7-2
- Recording to Multiple Files 7-4
- Pausing the Recording Process 7-6
- Stopping the Recording Process 7-6

Enabling Video Recording 7-7
- Recording Streaming Video from VR-Vantage 7-7
- Recording Streaming Video from a Webcam 7-8

Saving Files 7-9
- File Backups 7-9
- Saving a File to a New Name 7-9
- Saving a Portion of a File to a New File 7-10

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. If you want to record a video stream and video streaming is not enabled, click the Record From Stream button.
4. If video recording is enabled, optionally specify an alternative port, file name, or both.

The video stream must be a single video. The Logger does not support video multiplexing.

5. 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.

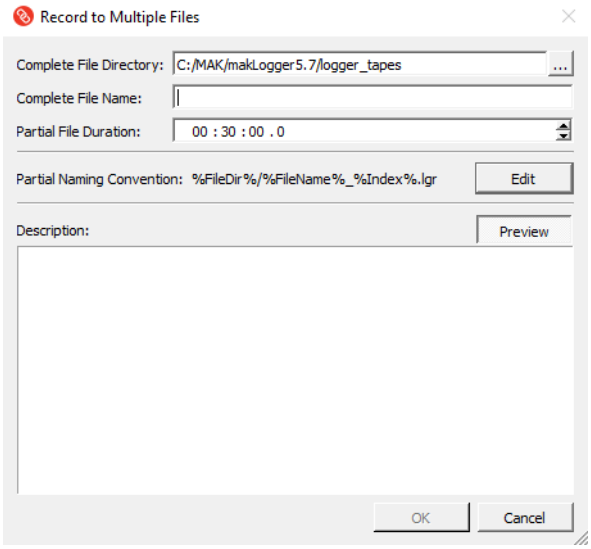


Figure 7-1. Record to Multiple Files dialog box

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_file-name_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).

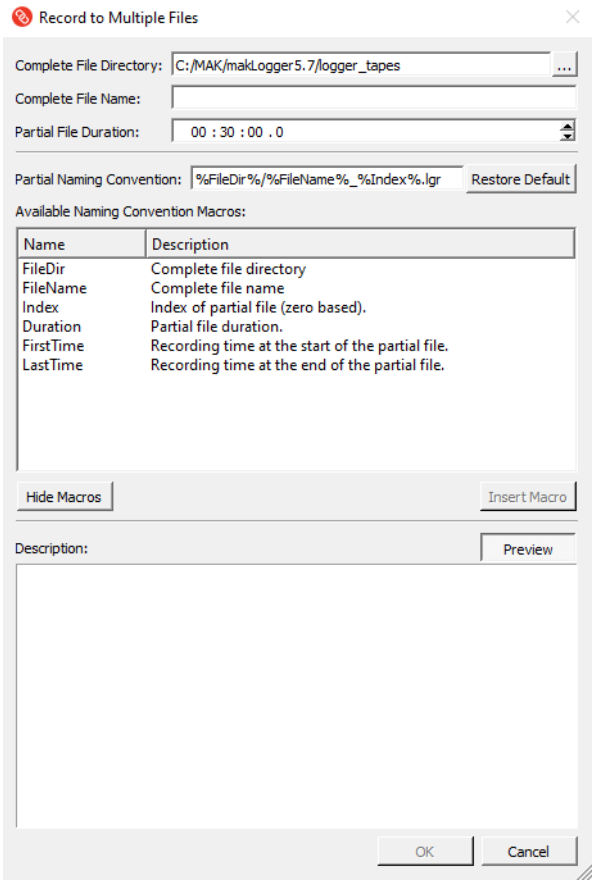


Figure 7-2. Record to Multiple Files dialog box in Edit mode

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, FirstTime, 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. Enabling Video Recording

By default, video recording is disabled. (You can change the default in the Preferences dialog box.) You can enable it in several ways.

To enable video recording, use any of the following methods:

- Choose **Record → Video Recording**.
- Click the Enable Recording button ( ) on the toolbar.
- Select the Record Videos check box on the Logger Preferences dialog box, Video Settings page.
- Click the Record from Stream button on the New Logger File dialog box.

- When you are recording video, the video stream is not displayed in the Video Stream View. However, an icon is added to the status bar to show that video is being recorded.
- The Logger does not record audio, except for audio that is sent using DIS/HLA PDUs, such as a signal PDU. The PDUs are recorded as part of the regular DIS/HLA data stream.

7.2.1. Recording Streaming Video from VR-Vantage

VR-Vantage, MAK's image generation application, can stream video from an observer window. (For details, please see *VR-Vantage Users Guide*.) To record the VR-Vantage video stream in the Logger, select the MPEG2 - VLC Out video stream. By default, VR-Vantage and the Logger both default to port 5000, so you can record a VR-Vantage video stream without changing the port in either product. As long as the Logger is configured to record the same port, it will record video directly from VR-Vantage, synchronized to whatever simulation data is being sent.

The Logger also supports using H264 for the video stream. To stream video from VR-Vantage using H264, the VR-Vantage window height and width must be multiples of 16 (for example, 1280 X 1024, 1024 X 768). Otherwise, the video stream will be unusable. For information about setting the VR-Vantage window size, please see *VR-Vantage Users Guide*.

7.2.2. Recording Streaming Video from a Webcam

The Logger includes a separate executable (*./bin64/webcamStream-Launcher.exe*) that can automatically activate the first webcam it finds on your computer and stream its video through the network. By default, the software is configured to use your loopback connection and port 5000, which is also the default port for the Logger.

The webcam executable also allows you to create your own GStreamer pipeline.

To stream webcam video:

1. On the Start menu, choose **MAK Technologies** → **Webcam Stream Launcher**, or run *./bin64/webcamStreamLauncher.exe*. Webcam Stream Launcher opens (Figure 7-3).

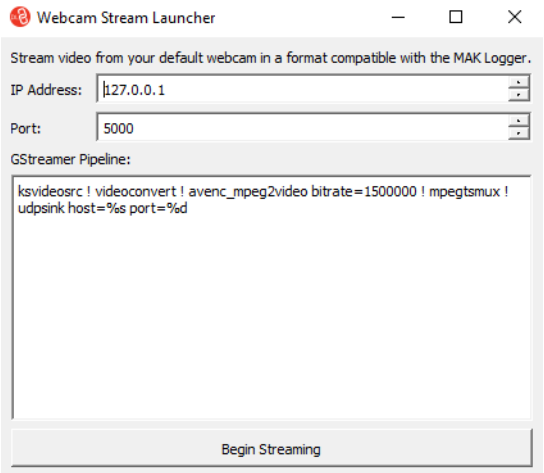


Figure 7-3. Webcam Stream Launcher

2. Optionally, edit the IP Address or Port.
3. Optionally, edit the GStreamer Pipeline.
4. Click Begin Streaming. As long as the Logger is configured to record from the same port, it will record video directly from your webcam, synchronized to whatever simulation data is being sent.

7.3. 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.

If video recording is enabled, when you save an unsaved file, the Logger creates a video file with the same name and the extension *.mp4*.

➤ 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).

7.3.1. 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 *brooklyndemoDIS.lgr* and save the changes, the original version of *brooklyndemoDIS.lgr* is saved to *brooklyndemoDIS.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 *brooklyndemoDIS.lgr* in the previous example, you were to edit it again and save it, a new version of *brooklyndemoDIS.bak.lgr* would replace the original backup file.

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

7.3.2. 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.

7.3.3. 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.

8. Playing Logger Files

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

Playing Logger Files 8-2
- Playing a Logger File 8-2
- Pausing Playback of a Logger File 8-2
- Looping Playback 8-2
- Stopping Playback of a Logger File 8-3
- Viewing Video 8-3
- Playing a Segment of a File 8-3
- Setting the Playback Speed 8-6
- Using High Speed Playback 8-6
- Playing a File While It Is Being Recorded 8-7

Zooming the Timeline 8-9
- Zooming the Timeline Using the Mouse 8-9
- Setting the Zoom from the Set Zoom Dialog Box 8-10
- Moving the Zoomed Selection 8-10

Moving the Timepoint to a Specific Time 8-11
- Jumping to the Beginning or End of the Playback Range 8-11
- Moving to a Specific Time in a File 8-11
- Setting the Time Using the Timeline Cursor 8-11

Viewing Packet Histograms 8-12
- Scaling the Packet Count View 8-13

Viewing Object Statistics 8-13

Viewing Streaming Video Using VLC 8-14
- Reducing Latency in VLC 8-14

8.1. 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 5-4.

8.1.1. 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**.

8.1.2. 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 ( ).

8.1.3. 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).

8.1.4. Stopping Playback of a Logger File

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

8.1.5. Viewing Video

When you open a recording that has a video file, you can enable or disable playback. (For details, please see "Opening Logger Files," on page 6-7.) This section describes the mechanics of viewing the video.

If a recording has a video file, when you open it, the Video Stream View opens automatically. When you start to play the recording, the video starts playing as well. If you want to see a larger view of the video, you can resize the Logger window or undock the Video Stream View and resize it.

If you want to hide the video, you can close the Video Stream View.

If you enabled video streaming when you opened the file, you can view the video in another application, such as VLC. (For information about viewing a video stream in VLC, please see "Viewing Streaming Video Using VLC," on page 8-14.) When you stream a video, an icon is added to the status bar to indicate that you are streaming video to the network.

8.1.6. 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 8-1) 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.

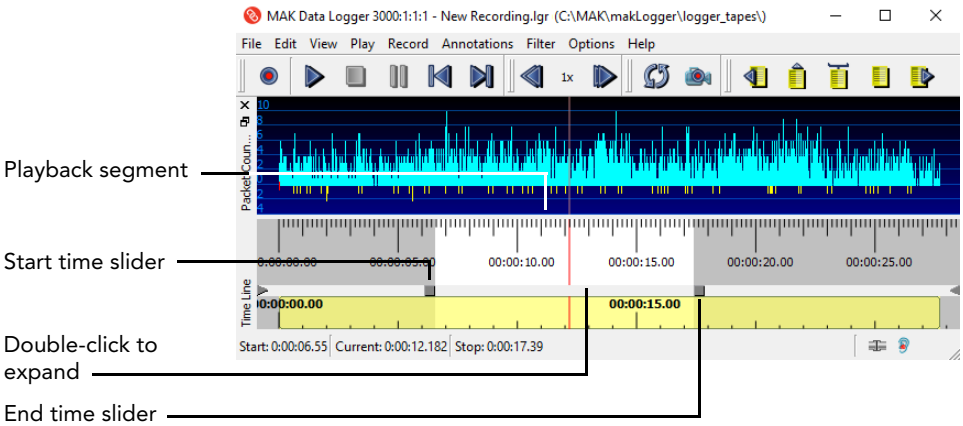


Figure 8-1. Timeline sliders

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 Times**. The Set Start and End Time dialog box opens (Figure 8-2).

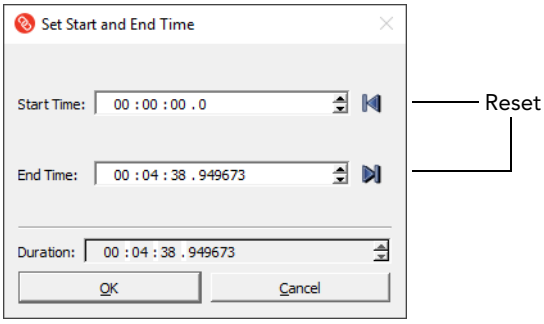


Figure 8-2. Set Start and End Time dialog box

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 8-1).

8.1.7. 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 8-3).

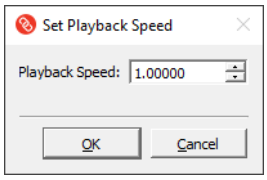


Figure 8-3. Set Playback Speed dialog box

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.

8.1.8. 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, edit the *lgrConfig.xml* file for your version of the Logger. 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 8-6, and if the speed meets or exceeds the threshold, high speed playback is used.

8.1.9. 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, this is *C:\Users\<user>\AppData\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.

8.2. 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 8-4 shows the zoom timeline zoomed to the range from approximately 00:45 to 2:37. 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.

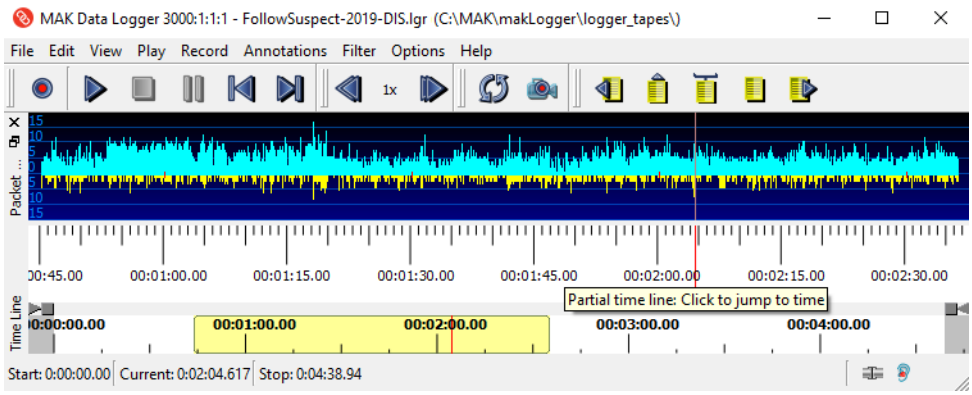


Figure 8-4. Zoomed timeline

8.2.1. 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.

8.2.2. 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 8-5).

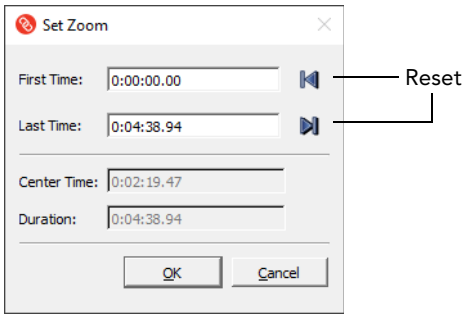


Figure 8-5. Set Zoom dialog box

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.

8.2.3. 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.

8.3. 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.

- 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.

8.3.1. 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 ( ).

8.3.2. 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 5-2.

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.

8.3.3. 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 oustide of the zoom range, click on the full timeline.

8.4. 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 8-6).

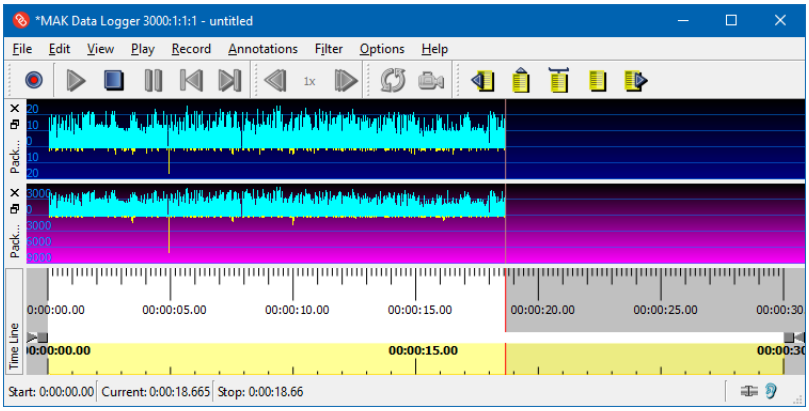


Figure 8-6. Packet histograms in record mode

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 14, *Using Logger Commands*.

- 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**.

8.4.1. Scaling the Packet Count View

You can change the scale of the Packet Count View (Figure 8-7). 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.

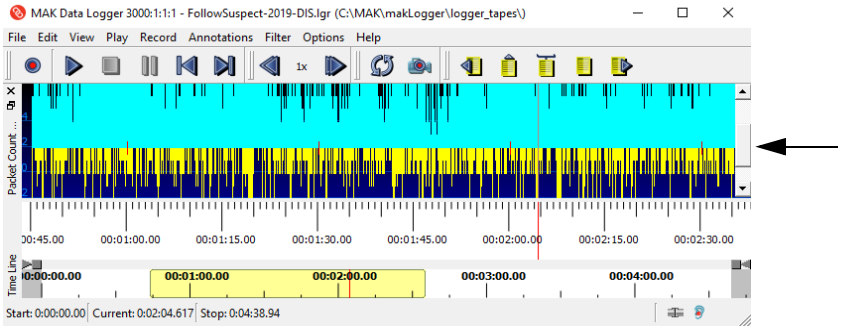


Figure 8-7. Scaled Packet Count View

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.

8.5. Viewing Object Statistics

The Logger can display statistics about the objects being published (during playback) or reflected (during recording) by the Logger application (Figure 8-8).

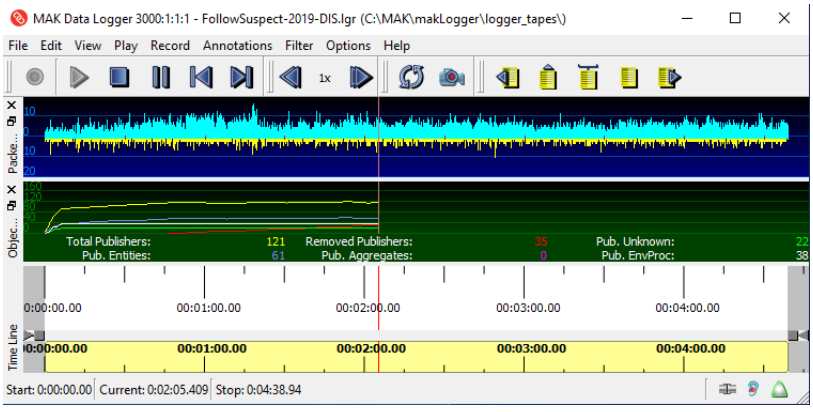


Figure 8-8. Object statistics

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

8.6. Viewing Streaming Video Using VLC

To view video streamed from the Logger in VLC:

1. Start the video stream.
2. Start VLC.
3. Choose **Media** → **Open Network Stream**.
4. For the network URL, put "udp://@:5000" (if the Logger's video playback port is 5000).

8.6.1. Reducing Latency in VLC

The default setting for viewing network streams in VLC includes a lot of buffering to make sure the video plays smoothly. This introduces a lot of latency on the receiving side. This section explains how to reduce that latency.

Reducing latency may lead to jitter. However, when working on a LAN, MAK engineers have rarely had problems.

To reduce latency when using VLC:

1. Start VLC.
2. Choose **Tools** → **Preferences**. The Preferences dialog box opens.
3. Select Input / Codecs.
4. In the Codecs group box, in the Hardware-accelerated Decoding list, select Automatic.
5. In the x264 Preset and Tuning Selection list, select ultrafast.
6. In the Network group box, in the Default Caching Policy list, select Lowest latency.
7. Click Save.

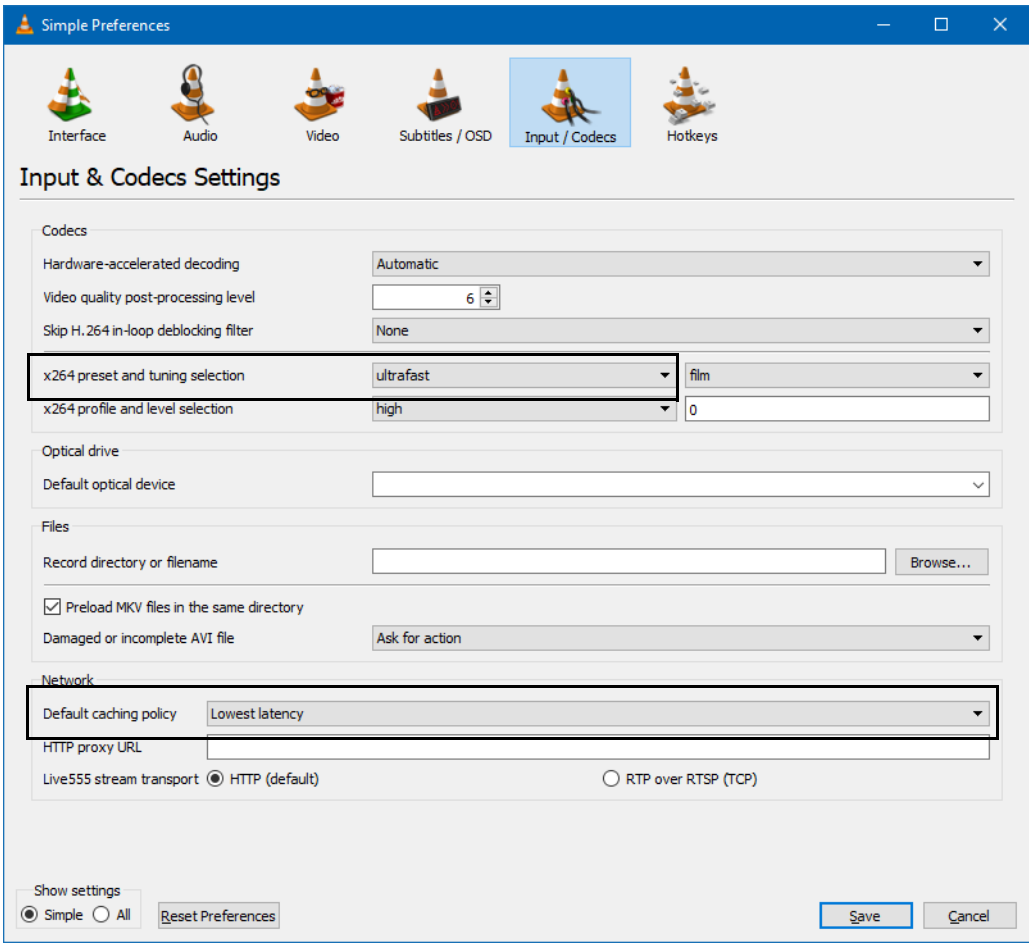


Figure 8-9. VLC Preferences, Inputs & Codecs

8. Choose **Media** → **Open Network Stream**. The Open Media dialog box opens.
9. Select the Network tab.
10. Enter the URL udp://@:5000 (or whatever port you have your system set up for).
11. Select the Show More Options check box. The dialog box expands with more options.
12. Set Caching to 1ms. (Do not set caching to zero. VLC will crash.)

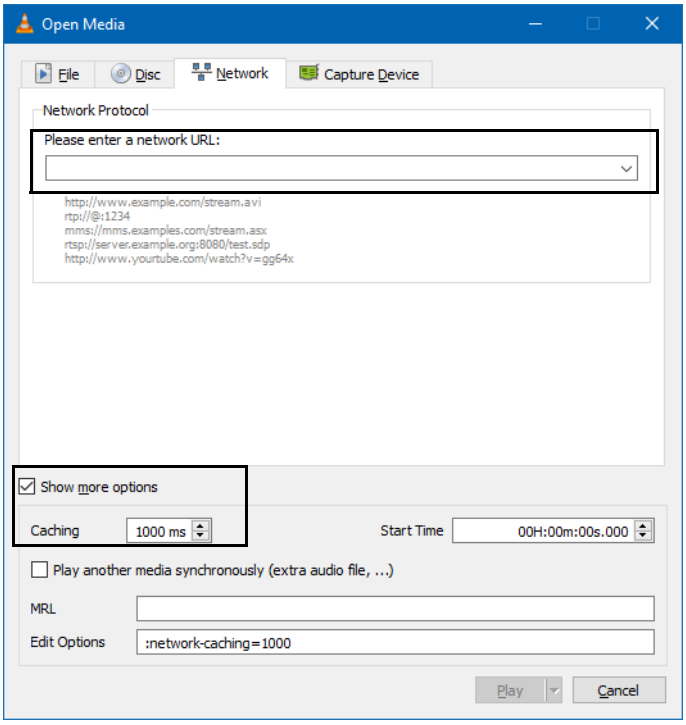


Figure 8-10. Open Media dialog box, Network tab

13. Click Play.

9. Annotating Files

This chapter explains how to add annotations to Logger files.

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

9.1. 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.

9.1.1. Adding an Annotation

To annotate a Logger file:

1. Choose **View** → **Annotations**. Annotations, if any, are displayed in the Logger data display area (Figure 9-1).

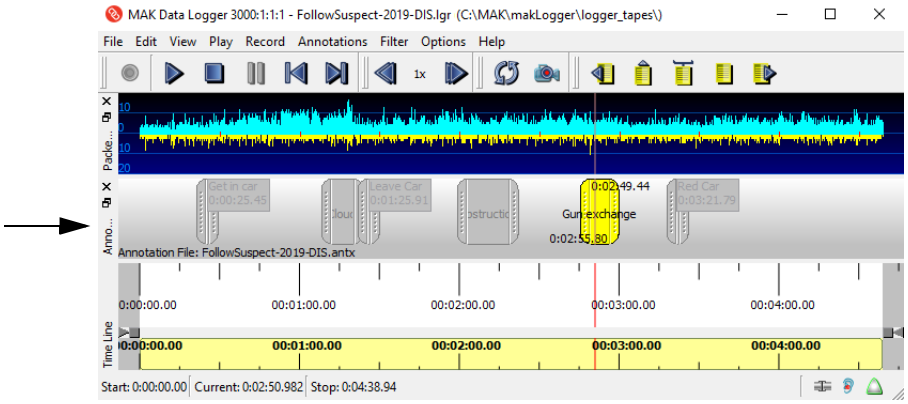


Figure 9-1. Annotations display

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 9-3.) The Point Annotation (or Range Annotation) dialog box opens.

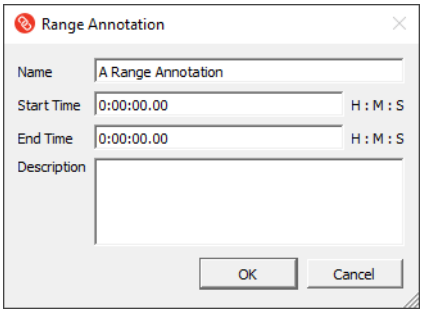


Figure 9-2. Range Annotation dialog box

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 9-1).

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 ( ).

9.1.2. 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 9-2.
2. Choose **Annotations** → **Save**. The Save Annotation File dialog box opens.
3. Type a name for the annotation file.
4. Click Save.

9.1.3. 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 loads 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.

9.1.4. 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.

9.1.5. 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.

9.1.6. 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 anyplace 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 9-5.

9.1.7. 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.

9.1.8. 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**.

9.1.9. 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 ( ).

9.1.10. Zooming the Timeline to an Annotation

You can zoom the timeline view to an annotation. For example, in Figure 9-3, in the first view, the zoom timeline shows the entire length of the file. After zooming to the Gun exchange 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.
- After zooming to an annotation, if you want to zoom out, you can use the mouse scroll wheel to zoom out or the Set Zoom dialog box (**View → Set Zoom**). You will have to manually move the left end of the yellow time line back to 0 or use the Set Zoom dialog box to reset it.

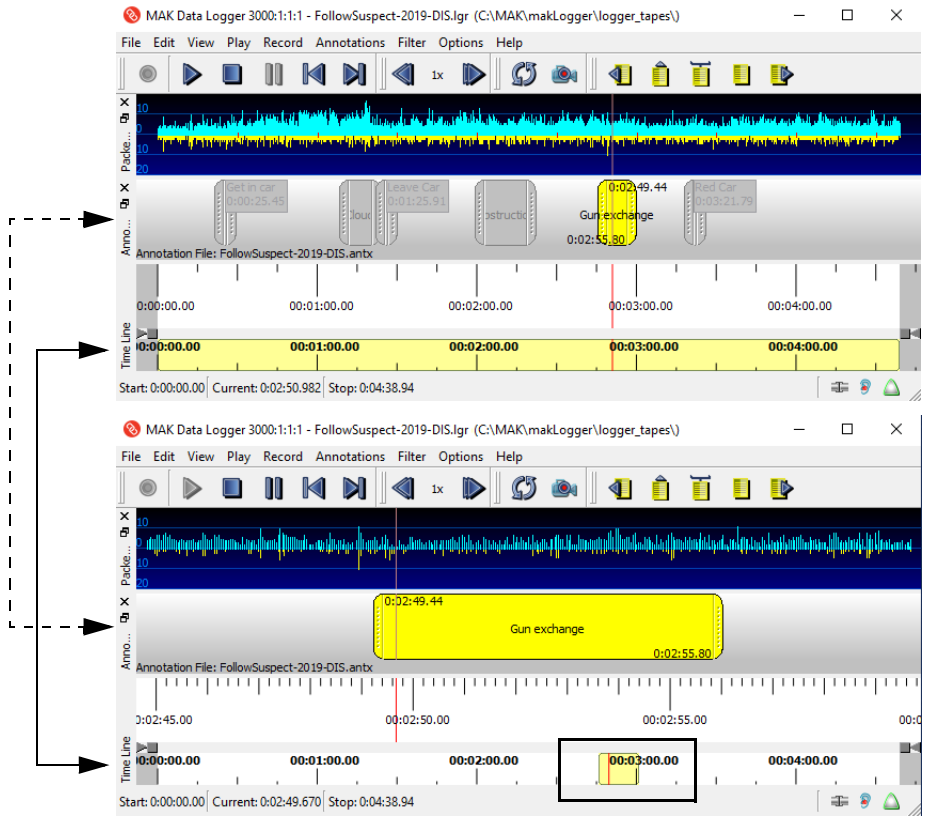


Figure 9-3. Zooming to an annotation

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 9-3).

9.1.11. Bounding Playback to an Annotation

You can set the playback segment to the range of an annotation. For example, in Figure 9-4, 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 9-3, the playback handles are not affected by zooming to the annotation.

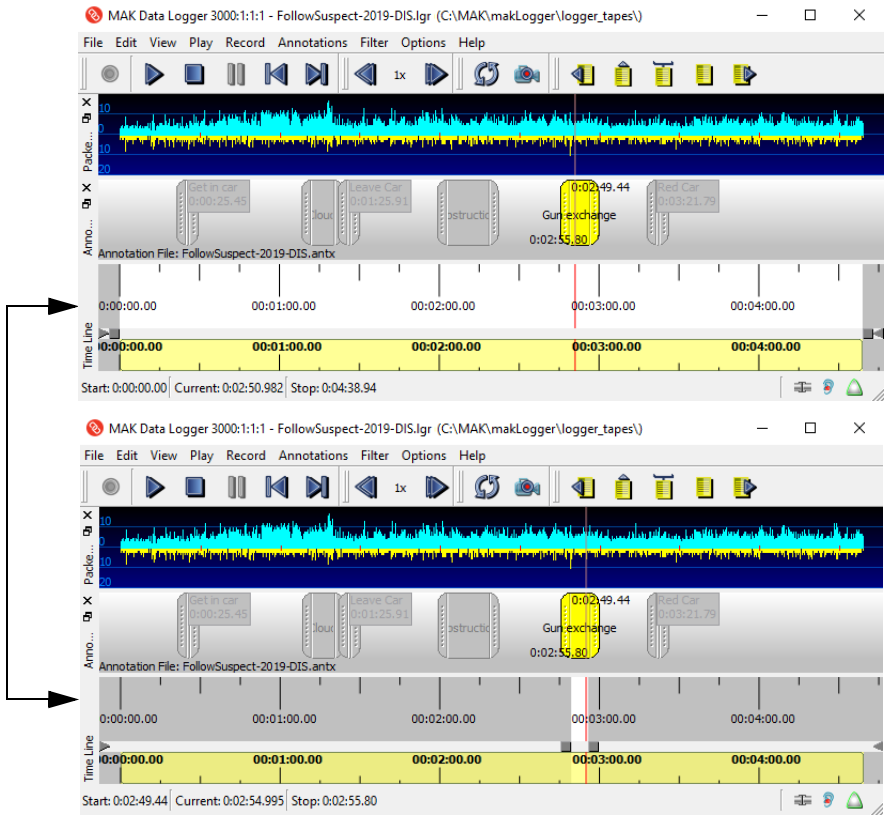


Figure 9-4. Bounding playback to an annotation

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 9-4).

10. 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 10-2
Filtering by Entity Type 10-3
Removing Entity Types from the Entity Types Lists 10-3
Configuring the List of Common Entity Types 10-4
Filtering DIS PDUs by PDU Kind 10-5
Including Custom PDUs in Filter Lists 10-6
Filtering DIS Messages by Entity ID 10-7
Filtering HLA Messages by Object Name 10-8
Filtering HLA Exercises by Class 10-10
Filtering by Marking Text 10-11
Filtering VR-Forces Messages 10-12

10.1. 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 entity type, class or object name. For both protocols, you can filter by marking text. During a Logger session, filtering settings apply to both recording and playback. Settings do not persist between sessions.

Besides configuring filtering in the GUI, as described in this chapter, you can configure filtering in the *lgrConfig* file by adding `listItem` elements to the various filter variables.

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

10.2. 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 10-1). 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.

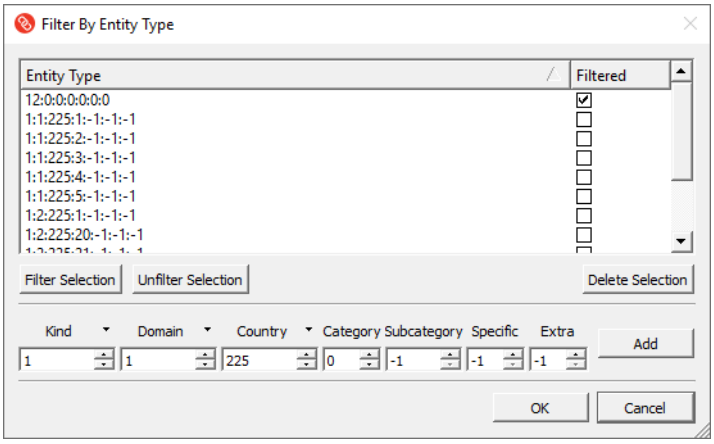


Figure 10-1. Filter By Entity Type dialog box

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.

10.2.1. 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.

10.2.2. 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 10-3. 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.

10.3. 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 10-2).

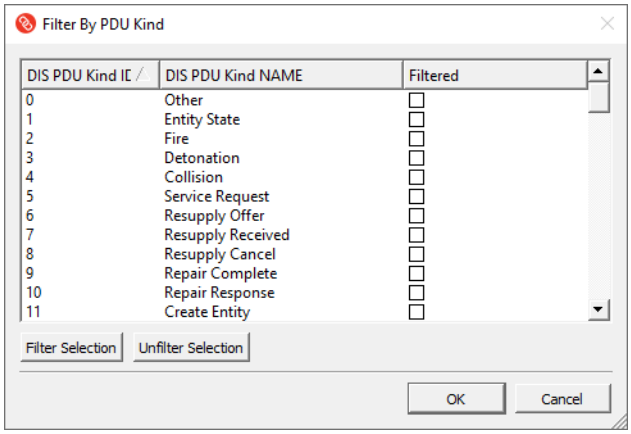


Figure 10-2. Filter By PDU Kind dialog box

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.

10.4. 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>
```

10.5. 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 10-3).

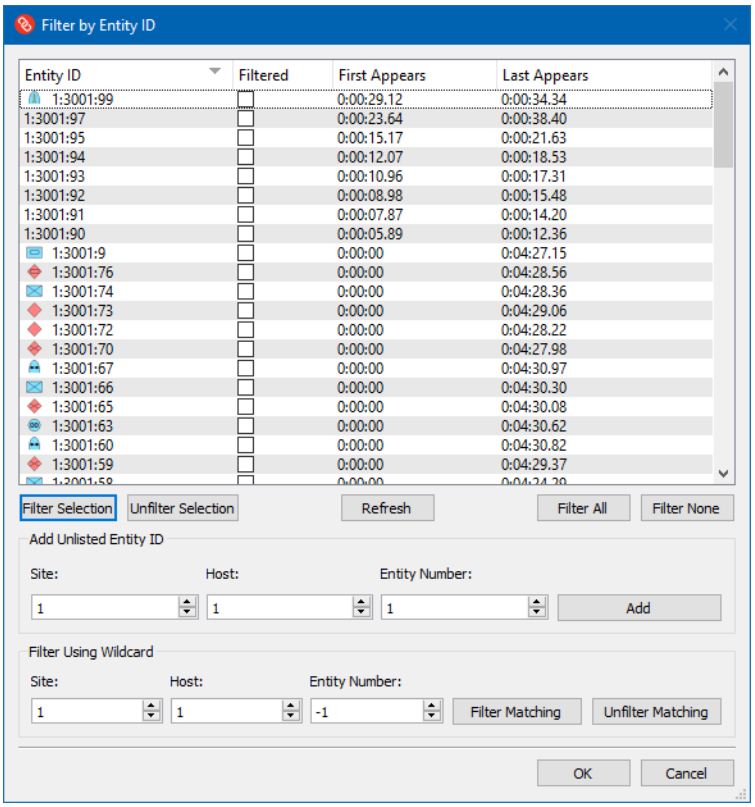


Figure 10-3. Filter by Entity ID dialog box

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.

10.6. 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 10-4).

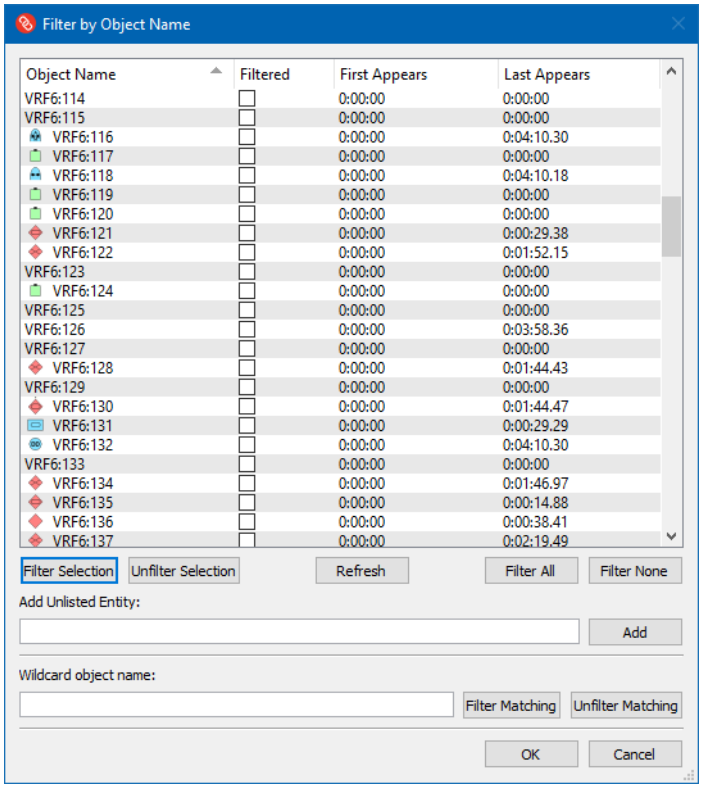


Figure 10-4. Filter by Object Name dialog box

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.

10.7. 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 10-5).

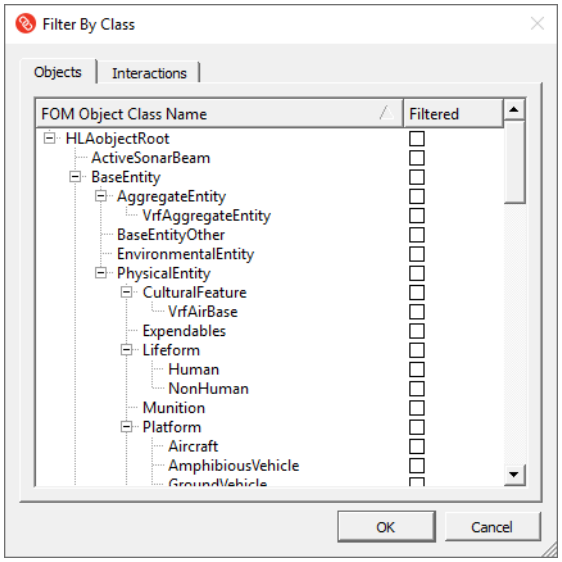


Figure 10-5. Filter By Class dialog box

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.

10.8. Filtering by Marking Text

You can filter messages by the marking text of the objects. You can filter specific names in the list or use wild cards. You can also add marking text

To filter by marking text:

1. Choose **Filter** → **By Marking**. The Filter By Marking dialog box opens (Figure 10-6).

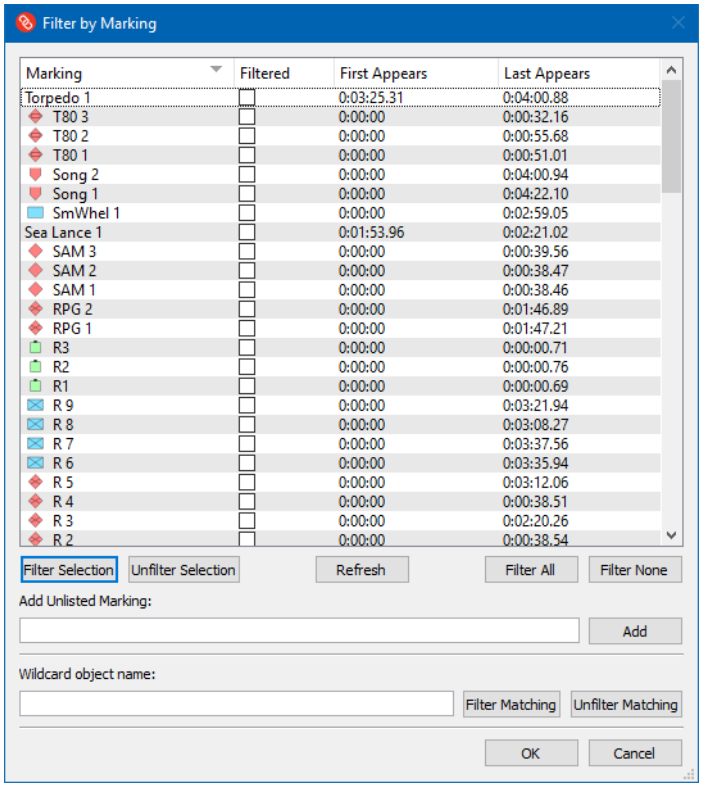


Figure 10-6. Filter By Marking dialog box

2. In the list of objects, select the ones that you want to filter or unfilter.
3. Click Filter Selection or Unfilter Selection.
4. To filter all objects, click Filter All.
5. To clear all selections, click Filter None.
6. If you want to filter marking text that is not present in the list, in the Add Unlisted Marking box, type the text and click Add.
7. If you want to filter using wildcards, in the Wildcard Object Name box, type the wildcard string. Then click Filter Matching or Unfilter Matching.
8. Click OK.

10.9. Filtering VR-Forces Messages

If you are recording a simulation that includes messages sent by VR-Forces, MAK'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 MAK 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 10-7).

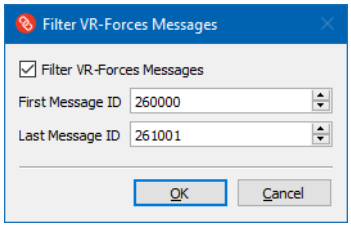


Figure 10-7. Filter VR-Forces Messages dialog box

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

11. Analyzing HLA Data

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

The HLA Traffic Analyzer 11-2
Traffic Analyzer Data Views 11-2
Hiding the Attributes Pane 11-3
How the Traffic Analyzer Evaluates Data 11-3
Representing Complex Datatypes 11-4
Datatype Representations Included in the Traffic Analyzer 11-4
Viewing Object and Interaction Classes and Attributes 11-5
Viewing the Attributes of an Object Class 11-5
Viewing the Parameters of an Interaction Class 11-5
Viewing Object Instances 11-6
Viewing Interaction Instances 11-7
Configuring the HLA Traffic Analyzer 11-9
Changing the Data Delimiter 11-10
Mapping Datatypes 11-11
Configuring Attribute Display Options 11-12
Mapping Entity Type Enumerations to Strings 11-12
Removing Brackets from the Display of Array Types 11-13

11.1. 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**.

11.2. 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 11-1 displays the Traffic Analyzer view.

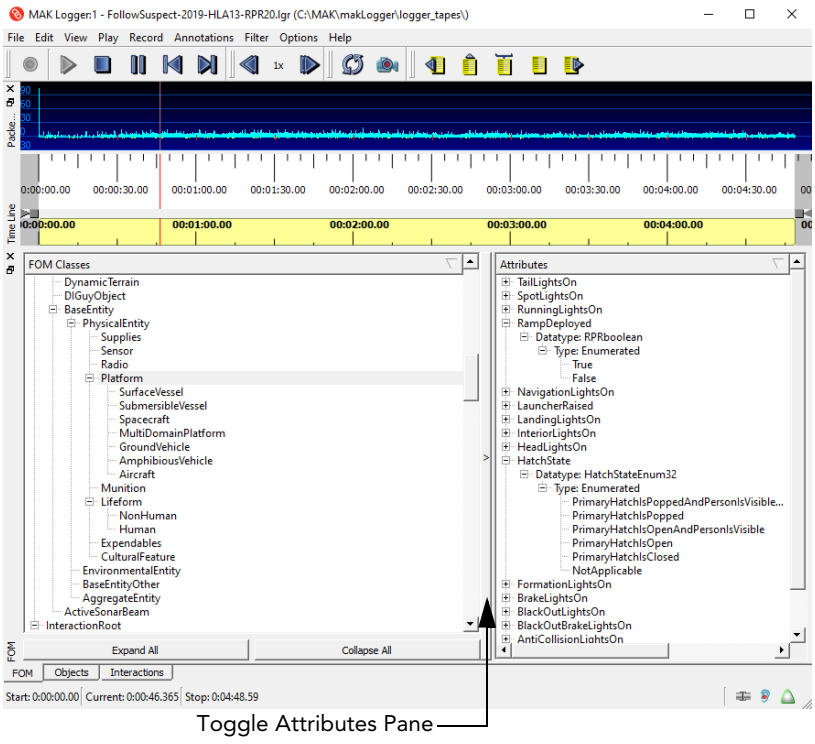


Figure 11-1. Traffic Analyzer, FOM tab

11.2.1. 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 11-1).

11.3. 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 11-4). 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 11-11.

11.3.1. 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 11-10).

11.3.2. 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.

11.4. Viewing Object and Interaction Classes and Attributes

The FOM tab (Figure 11-1) 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. 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.)

11.4.1. 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 11-1).

11.4.2. 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 11-1).

11.5. Viewing Object Instances

The Objects tab (Figure 11-2) of the HLA Traffic Analyzer 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.

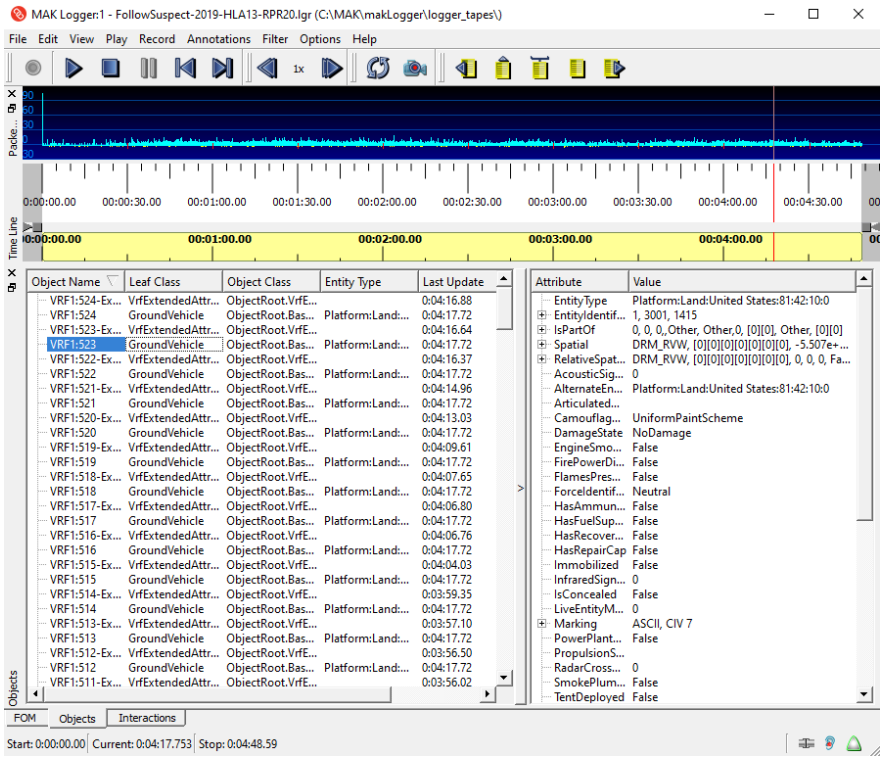


Figure 11-2. Objects tab

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.

11.6. Viewing Interaction Instances

The Interactions tab (Figure 11-3) of the HLA Traffic Analyzer displays the interactions in the HLA exercise and information about them.

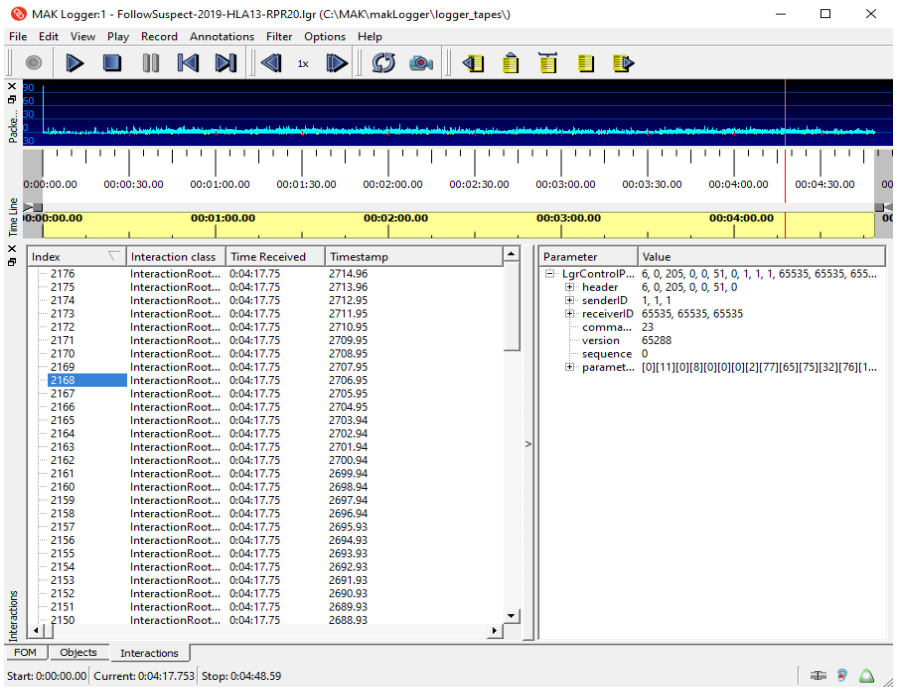


Figure 11-3. Interactions tab

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.

11.7. 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 11-4).

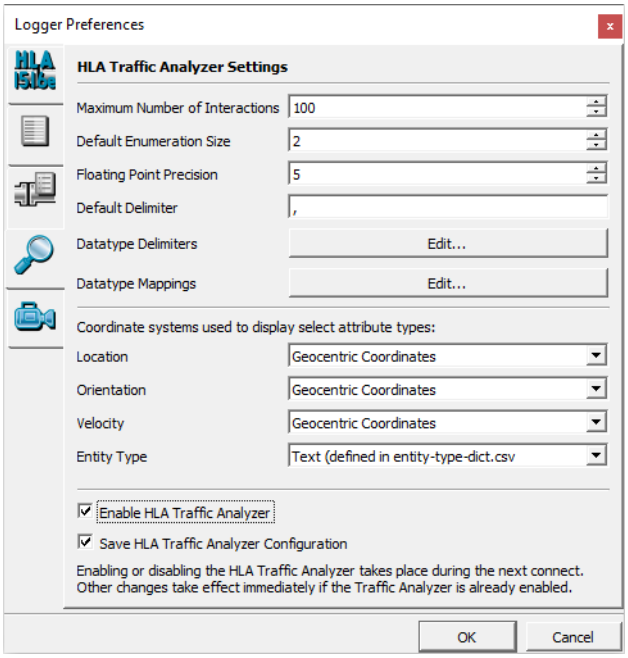


Figure 11-4. Logger Preferences dialog box, HLA Traffic Analyzer Settings

3. Change the values as desired. For details about changing delimiters and datatypes, please see "Mapping Datatypes," on page 11-11 and "Changing the Data Delimiter," on page 11-10.

11.7.1. Changing the Data Delimiter

As described in "Representing Complex Datatypes," on page 11-4, 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 11-4).
3. On the Datatype Delimiters line, click Edit. The Delimiter Mapper dialog box opens (Figure 11-5).

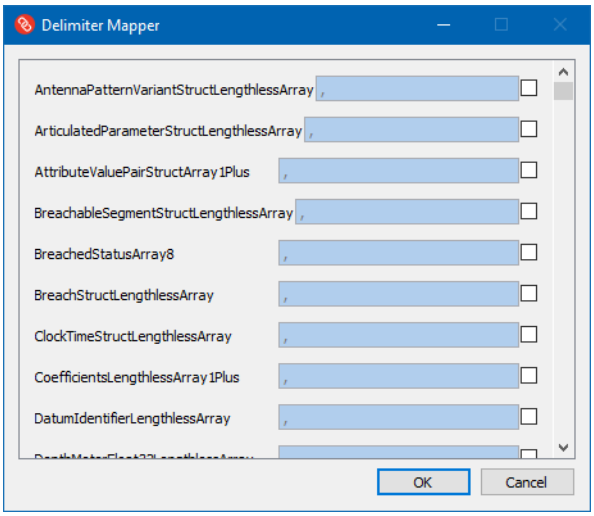


Figure 11-5. Delimiter Mapper dialog box

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.

11.7.2. 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 11-4).
3. On the Datatype Mappings line, click Edit. The Datatype Mapper dialog box opens (Figure 11-6).

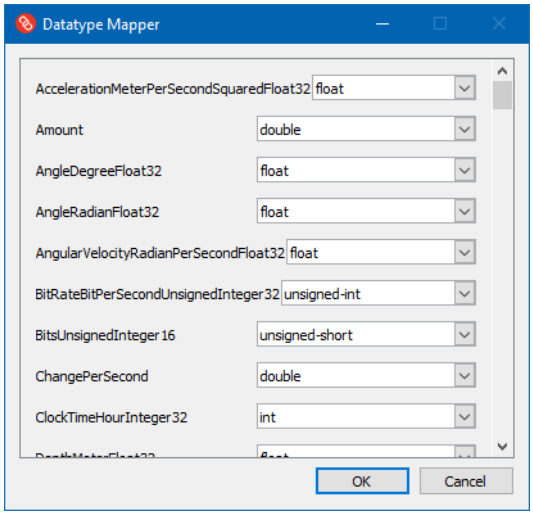


Figure 11-6. Datatype Mapper dialog box

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

11.7.3. 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.

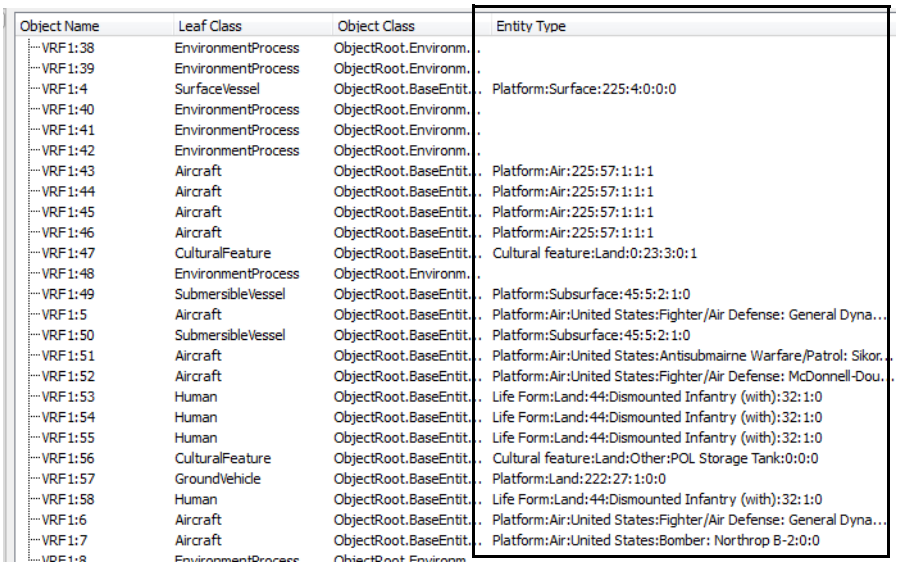
This option is only available for RPR FOM FOM mappers.

To specify entity display options:

1. Choose **Options** → **Preferences**. The Logger Preferences dialog box opens.
2. Select the HLA Traffic Analyzer Settings page (Figure 11-4).
3. For each attribute, choose an option from the list.
4. Click OK.

11.7.4. 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, *./bin64/entity-type-dict.csv* and *./bin64/SISO-REF-010.xml*.



Object Name	Leaf Class	Object Class	Entity Type
VRF1:38	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:39	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:4	SurfaceVessel	ObjectRoot.BaseEntit...	Platform:Surface:225:4:0:0:0
VRF1:40	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:41	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:42	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:43	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:225:57:1:1:1
VRF1:44	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:225:57:1:1:1
VRF1:45	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:225:57:1:1:1
VRF1:46	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:225:57:1:1:1
VRF1:47	CulturalFeature	ObjectRoot.BaseEntit...	Cultural feature:Land:0:23:3:0:1
VRF1:48	EnvironmentProcess	ObjectRoot.Environm...	
VRF1:49	SubmersibleVessel	ObjectRoot.BaseEntit...	Platform:Subsurface:45:5:2:1:0
VRF1:5	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:United States:Fighter/Air Defense: General Dyna...
VRF1:50	SubmersibleVessel	ObjectRoot.BaseEntit...	Platform:Subsurface:45:5:2:1:0
VRF1:51	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:United States:Antisubmairne Warfare/Patrol: Sikor...
VRF1:52	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:United States:Fighter/Air Defense: McDonnell-Dou...
VRF1:53	Human	ObjectRoot.BaseEntit...	Life Form:Land:44:Dismounted Infantry (with):32:1:0
VRF1:54	Human	ObjectRoot.BaseEntit...	Life Form:Land:44:Dismounted Infantry (with):32:1:0
VRF1:55	Human	ObjectRoot.BaseEntit...	Life Form:Land:44:Dismounted Infantry (with):32:1:0
VRF1:56	CulturalFeature	ObjectRoot.BaseEntit...	Cultural feature:Land:Other:POL Storage Tank:0:0:0
VRF1:57	GroundVehicle	ObjectRoot.BaseEntit...	Platform:Land:222:27:1:0:0
VRF1:58	Human	ObjectRoot.BaseEntit...	Life Form:Land:44:Dismounted Infantry (with):32:1:0
VRF1:6	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:United States:Fighter/Air Defense: General Dyna...
VRF1:7	Aircraft	ObjectRoot.BaseEntit...	Platform:Air:United States:Bomber: Northrop B-2:0:0

Figure 11-7. Entity type mappings

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

```
(entity-type-map-file "0SISO-REF-010.xml")
```

11.7.5. 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 11-8). 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)
```

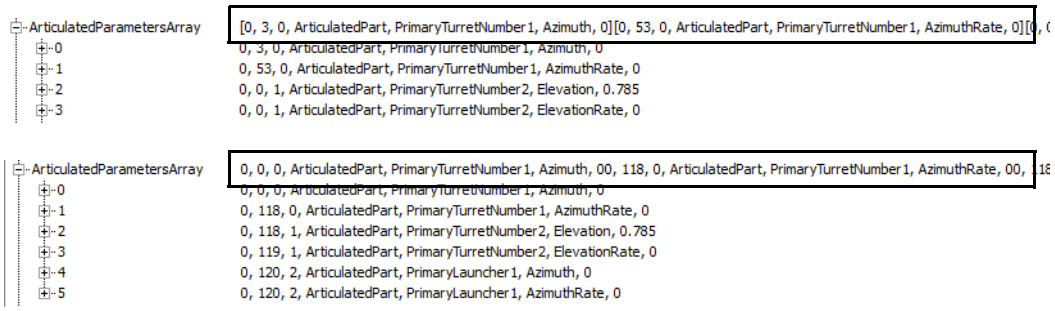


Figure 11-8. Array data, with and without brackets

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")
```

12. Editing Logger Files

This chapter explains how to merge, concatenate, and edit Logger files.

Editing Files 12-2
Reverting to the Saved Version of a File 12-2
Clearing and Deleting Segments of a File 12-3
Merging Files 12-4
Concatenating Files 12-5
Trimming Files 12-5
Offsetting Files 12-5

12.1. 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.
- If you edit a file that has a video associated with it, the video does not get edited. Depending on the type of editing you do, after editing, the data stream and video stream might not be correlated any more.

12.1.1. 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**.

12.2. 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 time-stamps 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 12-1 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.

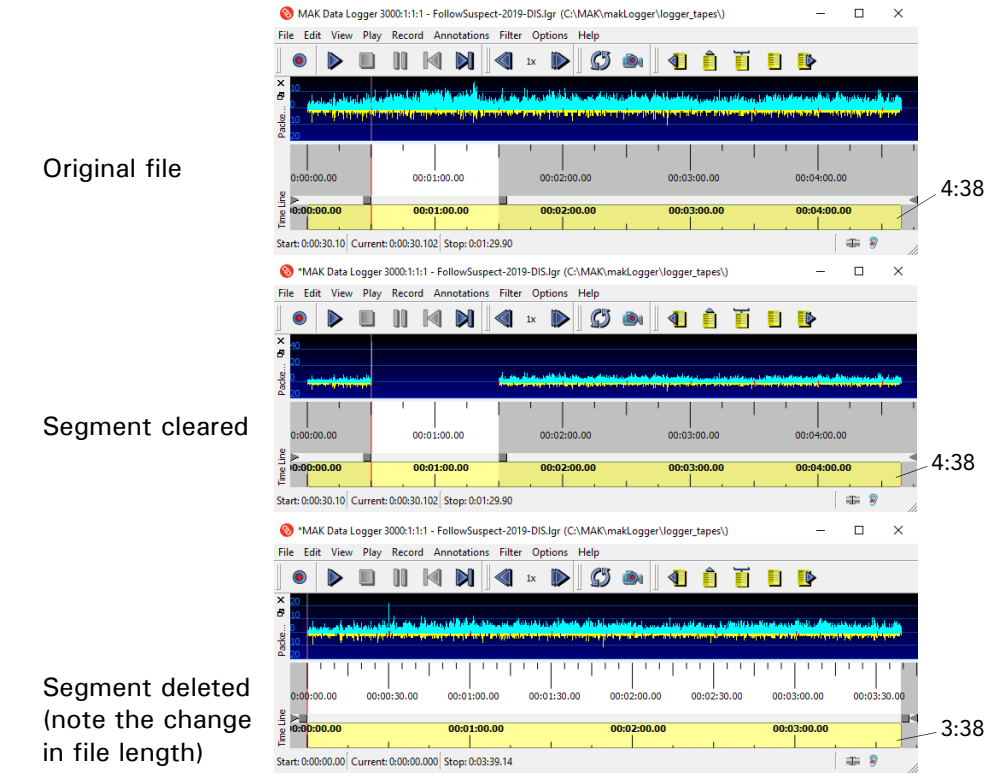


Figure 12-1. Clearing and deleting file segments

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.
- If you delete a section of a recording that has an initial state packet, the initial state information is lost.

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.

12.3. 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.

Recordings that are merged or concatenated combine their initial state packets. During playback the initial state packets are processed to appropriately add the entities at the time each initial state is processed.

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.

12.4. 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.

12.5. 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.

12.6. 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.

13. Exporting Logger Data

This chapter explains how to export Logger files to various data formats.

Exporting Data 13-2
Understanding Exported Data 13-3
Exporting Data to a Database 13-5
Configuring an ODBC Data Source 13-8
Exporting DIS Data to a Database 13-9
Exporting HLA Data to a Database 13-10
The Export Log 13-10
Configuring a Data Source for Microsoft Access 13-11
Installing MySQL and Configuring a Data Source 13-14
Installing MySQL and the MySQL ODBC Driver 13-14
Creating a MySQL Database 13-15
Configuring a Data Source for MySQL 13-15
Exporting Data on Linux 13-18
Importing Logger Data into a Spreadsheet 13-19
Importing Data from MySQL to Microsoft Excel 13-19
Importing Data from Access to Excel 13-22

13.1. 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 13-8, "Configuring a Data Source for Microsoft Access," on page 13-11, and "Installing MySQL and Configuring a Data Source," on page 13-14.
- The Logger does not apply filters when you export data.
- When you export data, you can have the data resampled with the current checkpoint policy. For more information, please see "Configuring Indexing," on page 5-2.

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.

13.2. Understanding Exported Data

The Logger records the data sent over the network using the DIS or HLA protocols. The HLA protocol is further defined by a Federation Object Model. By default, MAK applications use the Real-Time Platform Reference FOM (RPR FOM) for HLA. For details about DIS and HLA, review the IEEE and SISO standards to understand how the simulation data is sent using these protocols. If you are using HLA, you can probably get most of the information you need from the SISO RPR FOM document. The DIS standard provides more detail about the data content and when and why data is sent. Standards are available at:

- https://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=30822&PortalId=0&TabId=105
- https://standards.ieee.org/findstds/standard/1278.1-2012.html
- http://standards.ieee.org/develop/wg/1516_WG.html (primarily you want to look at the OMT)

When the Logger records a DIS exercise, it creates a separate packet for each DIS PDU it receives. The Logger exports the packet data into tables by the PDU type (that is, a different table for each type of PDU). There are columns for:

- The PDU data fields.
- A unique key (per table).
- Packet arrival time (relative to the start of the file at 0.0).
- DIS time stamp.
- Time stamp type.
- Exercise ID.
- PDU size.

Data types that contain an array of data are broken out into subtables (for example, articulated and attached parts). The subtable entries contain a key to link them back to the specific PDU entry.

In DIS, to find out when a particular attribute of an entity changed, you must search through the records for that entity and determine when the value changed. Each entry contains the full contents of the PDU.

When the Logger records an HLA exercise, the HLA classes are divided into Object Classes (state-full data) and Interaction classes (transient data). The Logger creates a separate packet for each Interaction (usually all parameters) and Object update (usually just a subset of attributes of changed data, except the first one or two updates for an object) it receives. The Logger exports the data into tables by HLA class.

The Logger exports objects into an object class table format. It exports interactions into an interaction class table format. Interaction tables have columns for each parameter of the Interaction along with a unique key (per table), the arrival time, and the user tag. Objects have an object instance table for all object instances (of all classes) and an update table for each class of object. The object instance table has:

- A unique key (per table).
- Arrival time.
- Object name.
- Object handle.
- Object class name.
- Object class handle.
- Object discovery time.
- Object removal time.

The object update table has columns for each attribute in the object class and a key back to the object instance table to which the update belongs. Again, data types that contain an array of data (or that contain many sub fields) are broken out into subtables with a key to link them back to the parent entry.

In HLA, to find out when a particular attribute of an entity changed, you must search through the records for that entity and determine when the value changed. Unlike DIS, most object updates only send changed data (usually location, velocity, acceleration and so on). The record contains null values for those attributes that were not updated. When you look for changes, consider the value NULL as a non-change to the previous non-NULL value. You could look for when NULL changes to non-NULL, but sometimes data that has not changed is transmitted over the network (for example, in response to a request attribute update).

For both DIS and HLA, the Logger does not clear out a database when exporting. Its export operations are additive.

13.3. Exporting Data to a Database

The Logger can export data to a database using ODBC. Please see *MAK 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 13-8, "Configuring a Data Source for Microsoft Access," on page 13-11, and "Installing MySQL and Configuring a Data Source," on page 13-14.
- 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 13-11.
- 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 13-1 and Figure 13-2).

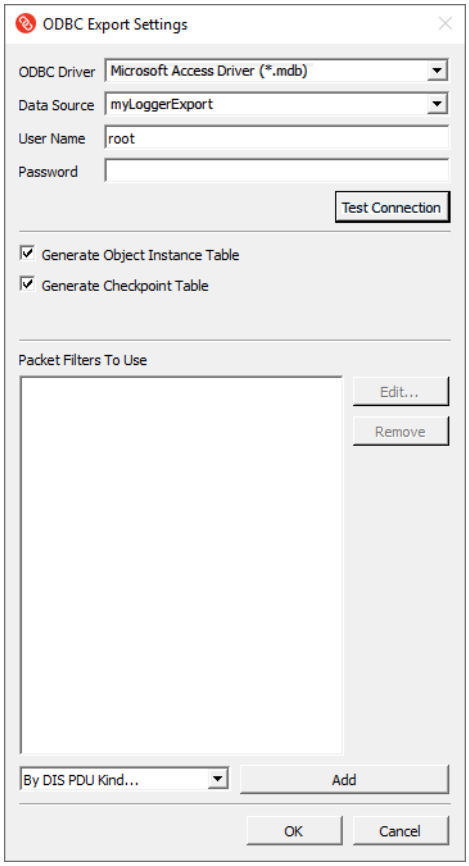
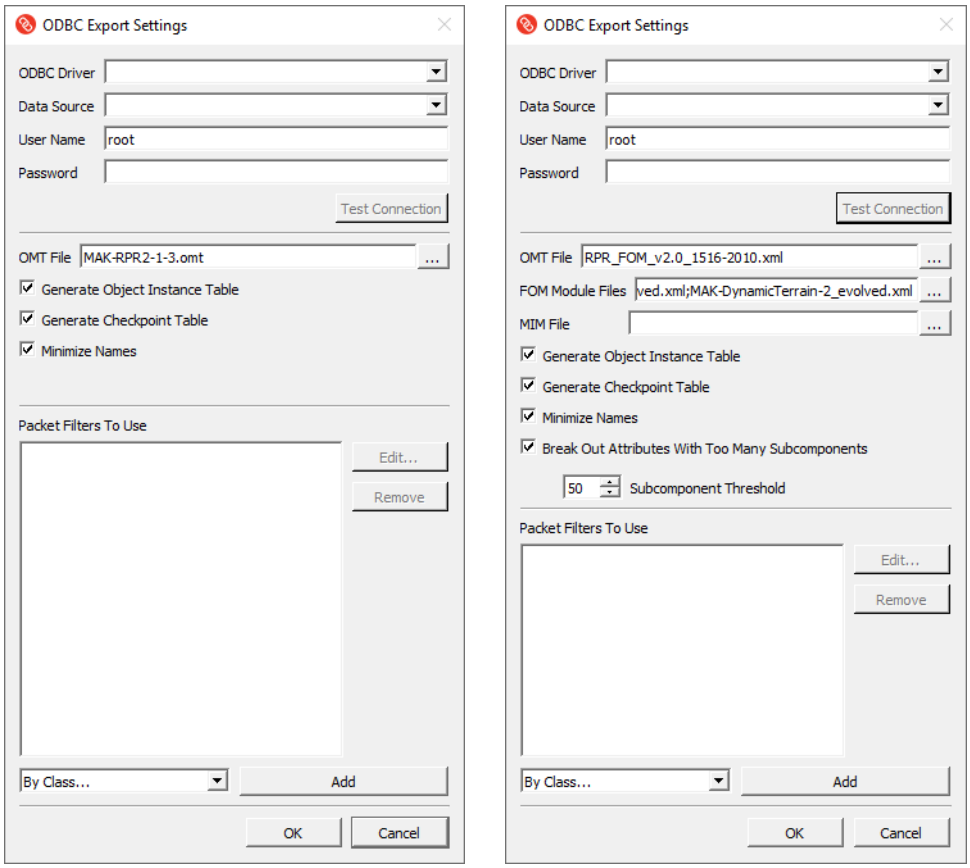


Figure 13-1. ODBC Export Settings dialog box (DIS)



HLA 1.3 and 1516 HLA Evolved

Figure 13-2. 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, select 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 13-10.)

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 10-2.
12. Click OK.

13.3.1. 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.

13.3.2. 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.

13.3.3. 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.

13.3.4. The Export Log

When you export Logger data, the Logger writes information about the export process to *sql_export.txt* in the *./bin64* directory from which you are running the Logger. The log file is important because it identifies the data that cannot be exported.

13.4. 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. You must install the 64-bit version of MS 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 13-22.)

! 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. To open the ODBC Data Source Administrator dialog box, first open the Control Panel.
2. Select Administrative Tools. (You might need to select System and Security first.)
3. Select Data Sources (ODBC) or ODBC Data Sources (64-bit). The ODBC Data Source Administrator dialog box opens (Figure 13-3).

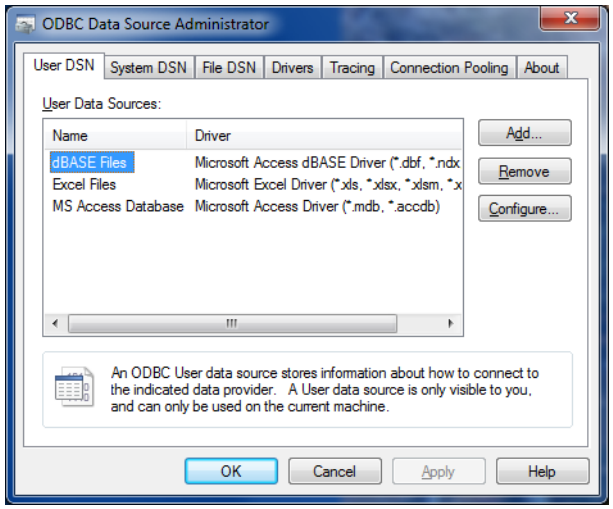


Figure 13-3. ODBC Data Sources Administrator

4. 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.
5. Click Add. The Create New Data Source window opens (Figure 13-4).

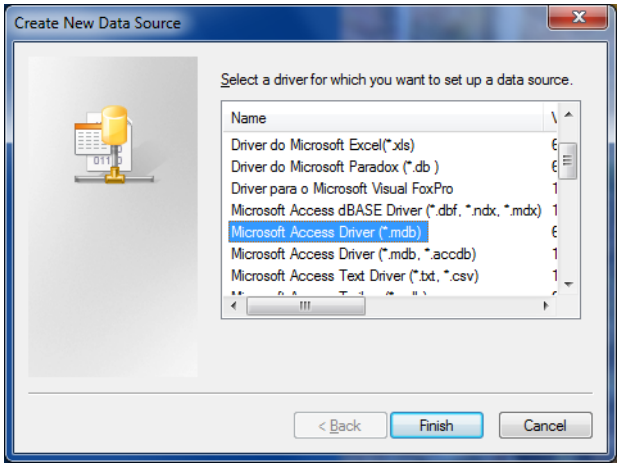


Figure 13-4. Create New Data Source window

6. Select Microsoft Access Driver (*.mdb). (The Logger looks for a driver with this exact text. Any other driver will not be recognized.)
7. Click Finish. The ODBC Microsoft Access Setup dialog box opens (Figure 13-5).

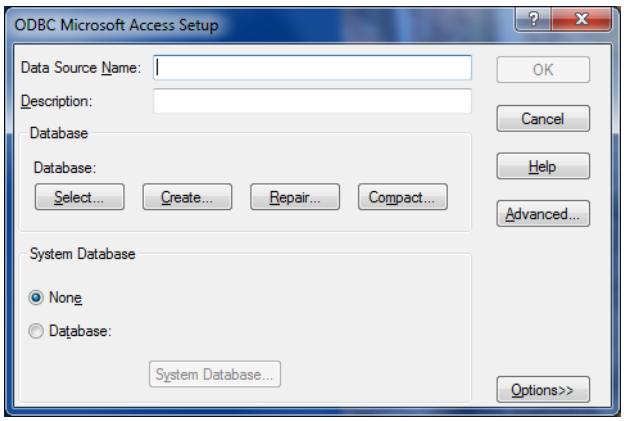


Figure 13-5. ODBC Microsoft Access Setup dialog box

8. In the Data Source Name box, type a name for the data source.
9. In the Database group box, click Create. The New Database dialog box opens (Figure 13-6).

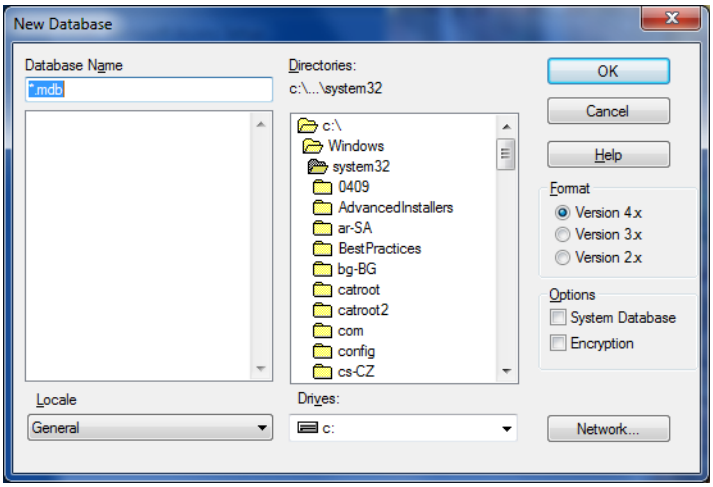


Figure 13-6. New Database dialog box

10. In the Database Name box, type a name for the database, for example, *myNewLoggerExport.mdb*.
11. In the Directories list, select the directory where you want to create the database (Figure 13-7).

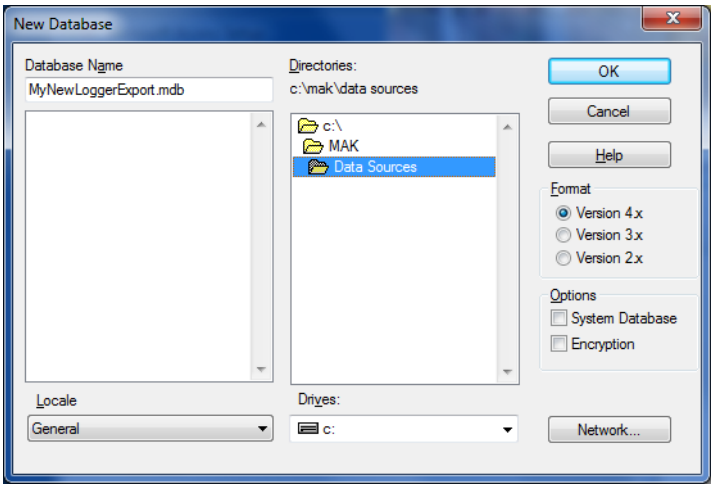


Figure 13-7. Completed New Database configuration

12. Click OK. An informational prompt advises you that the database has been created (Figure 13-8).

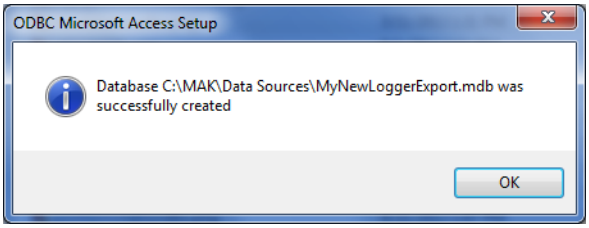


Figure 13-8. Information prompt

13. Click OK.
14. 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.

13.5. 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 *MAK Data Logger Release Notes* for a list of supported versions of MySQL. You must use a supported version.

! The 64 bit version of the Logger can only export to data sources created with a 64 bit version of Microsoft Access or MySQL.

13.5.1. 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.

13.5.2. Creating a MySQL Database

You must create a database before you can set up a data source.

To create a MySQL database:

1. On the Start menu, choose **MySQL → 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
   ```

13.5.3. 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 *MAK 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.
2. Open the Control Panel.
3. Select System and Security.
4. Select Administrative Tools.
5. Select Data Sources (ODBC). The ODBC Data Source Administrator dialog box opens (Figure 13-9).

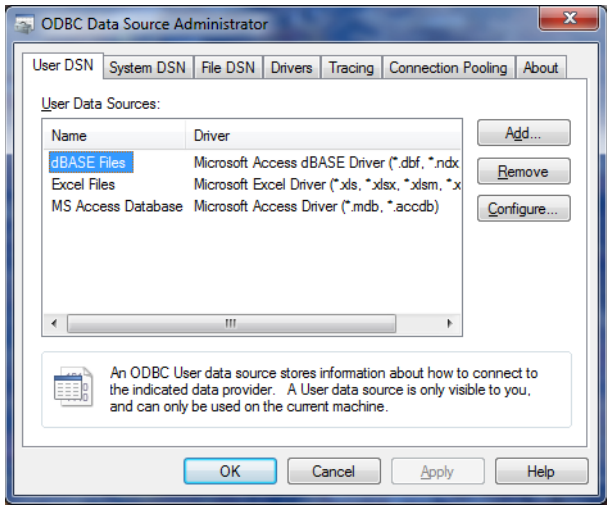


Figure 13-9. ODBC Data Source Administrator dialog box

6. Click Add. The Create New Data Source dialog box opens (Figure 13-10).

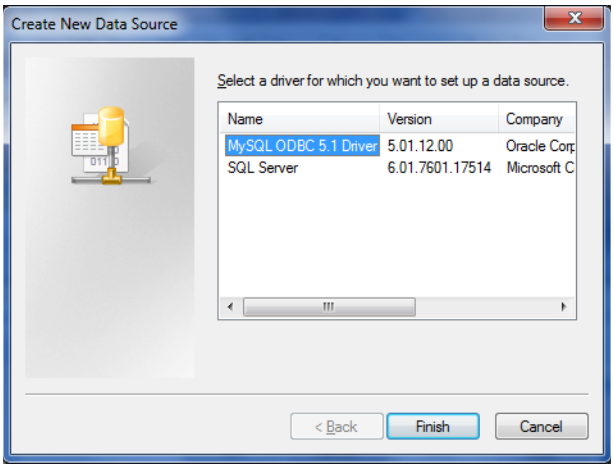


Figure 13-10. Create New Data Source dialog box

7. Select the MySQL ODBC driver.
8. Click Finish. The ODBC Data Source Configuration dialog box opens.

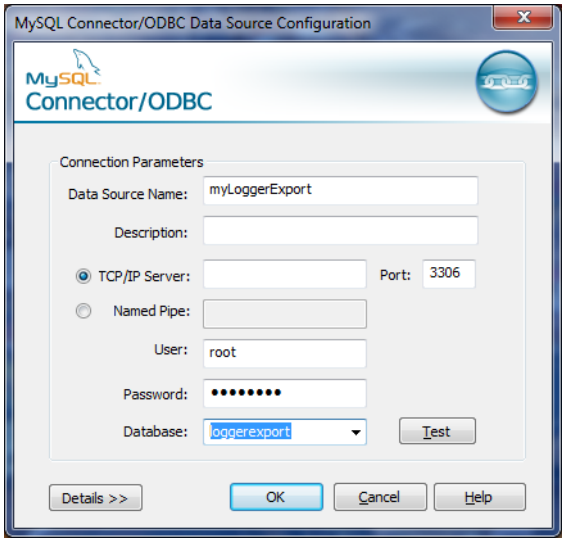


Figure 13-11. ODBC Data Source Configuration dialog box

9. Type a name for your data source.
10. Complete the Server, User, and Password fields.
11. Enter the name of the database that you created for Logger data.
12. Click OK. The new data source is displayed in the ODBC Data Source Administrator dialog box (Figure 13-12).

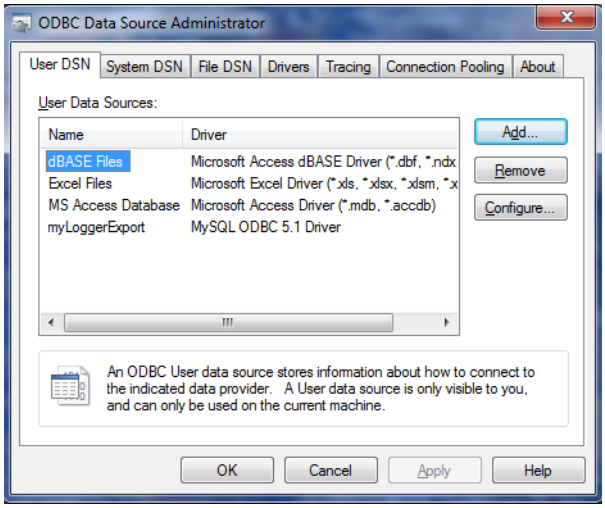


Figure 13-12. ODBC Data Source Administrator dialog box

13.6. Exporting Data on Linux

To export Logger data on Linux, you must:

- Install the ODBC driver.
- Set up MySQL.
- Create a database.
- Create and edit *.iodbc.ini*.

Install the ODBC Driver

1. Download *libiodbc-3.52.12.tar.gz* at http://www.iodbc.org/dataspace/doc/iodbc/wiki/iodbcWiki/Downloads.
2. Configure, build and install it.

Assuming you use the defaults, you now have */usr/local/lib/libiodbc.so*.

Install MySQL

1. Download the MySQL 5.3.10 ODBC connector for RH6 (64 bit) (*mysql-connector-odbc-5.3.10-1.el6.x86_64.rpm*) at https://dev.mysql.com/downloads/connector/odbc/
2. Install the RPM.

From this we will use */usr/lib64/libmyodbc5a.so* (MySQL ODBC Connector 5.3 ANSI driver).

Install and Set Up MySQL

1. Install MySQL Server using yum:

   ```
   yum install mysql-server
   ```

2. Start the server:

   ```
   service mysqld start
   ```

3. Create a database:

   ```
   mysql
   create database LoggerExport;
   ```

4. In your user's home directory, create *.iodbc.ini*. Populate it with the following (assumes the previous steps and default values):

   ```
   [ODBC Data Sources]
   myDataSource = MySQL ODBC 5.3 ANSI Driver
   [myDataSource]
   Driver = /usr/lib64/libmyodbc5a.so
   Description = MySQL ODBC 5.3 ANSI Driver
   Server = localhost
   Port =
   USER = root
   Password =
   Database = LoggerExport
   Option = 3
   ```

```
Socket = /var/lib/mysql/mysql.sock
```

Run MAK Data Logger

1. Set LD_LIBRARY_PATH with the following command:

   ```
   export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
   ```

2. Run the Logger. The **Export** → **Database** menu command should be available and things should be filled out to be able to export.

13.7. 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.

13.7.1. 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 13-13).

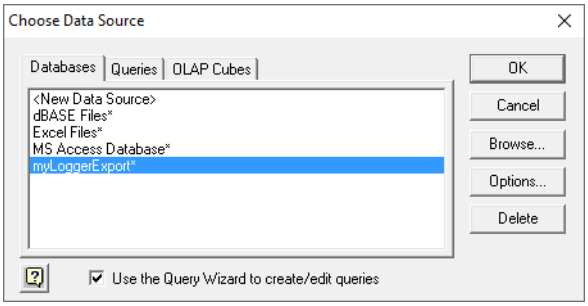


Figure 13-13. Choose Data Source dialog box

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 13-14). 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.

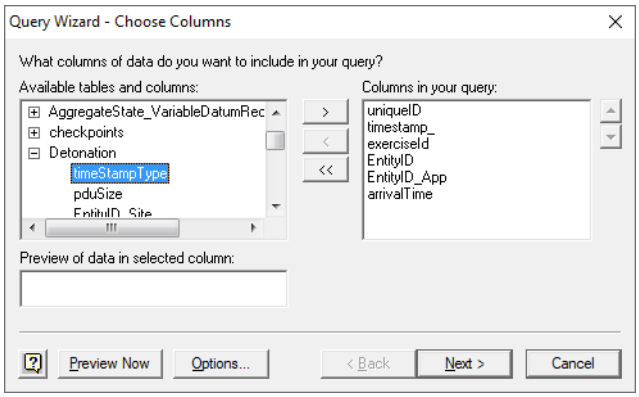


Figure 13-14. Query Wizard - Choose Columns

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 13-15).

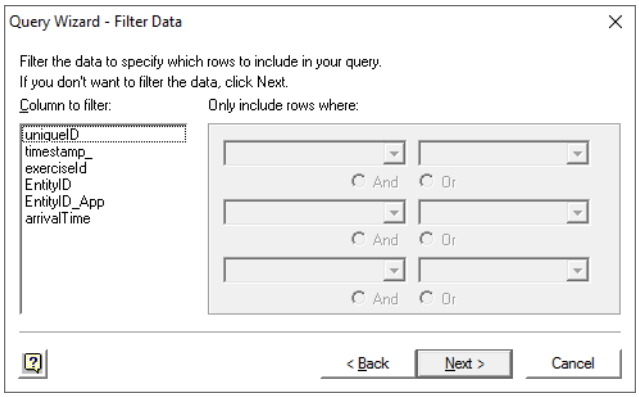


Figure 13-15. Query Wizard - Filter Data

8. Set up any filtering that you want, then click Next. The Query Wizard - Sort Order dialog box opens (Figure 13-16).

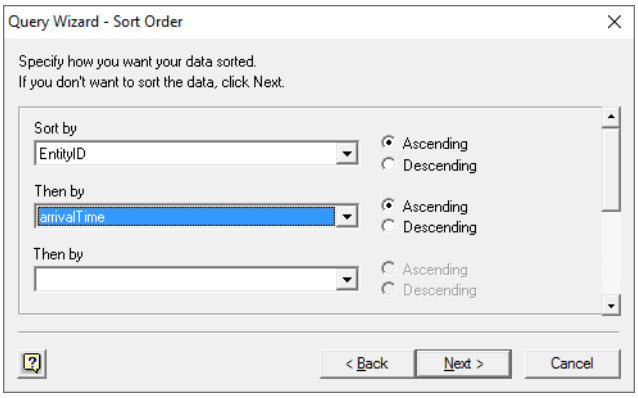


Figure 13-16. Query Wizard - Sort Order

9. Set up any sorting that you want, then click Next. The Query Wizard - Finish dialog box opens (Figure 13-17).

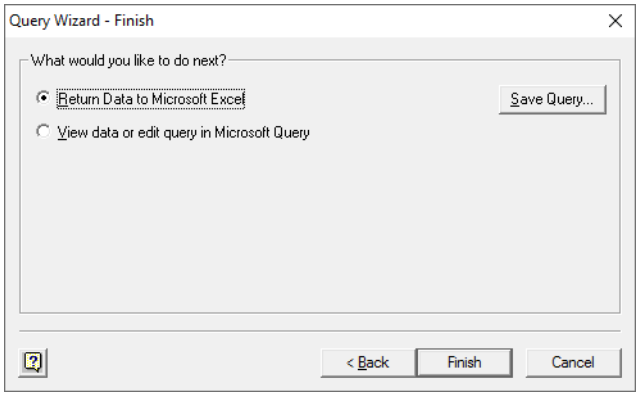


Figure 13-17. Query Wizard - Finish

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 13-18).

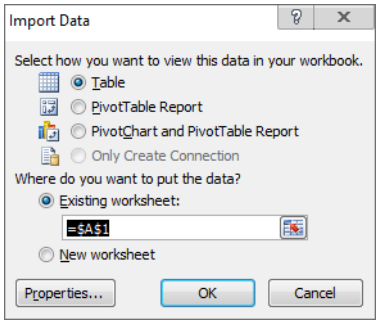


Figure 13-18. Import Data dialog box

11. Specify where you want the data to be imported to and click OK. The data is imported into the spreadsheet (Figure 13-19).

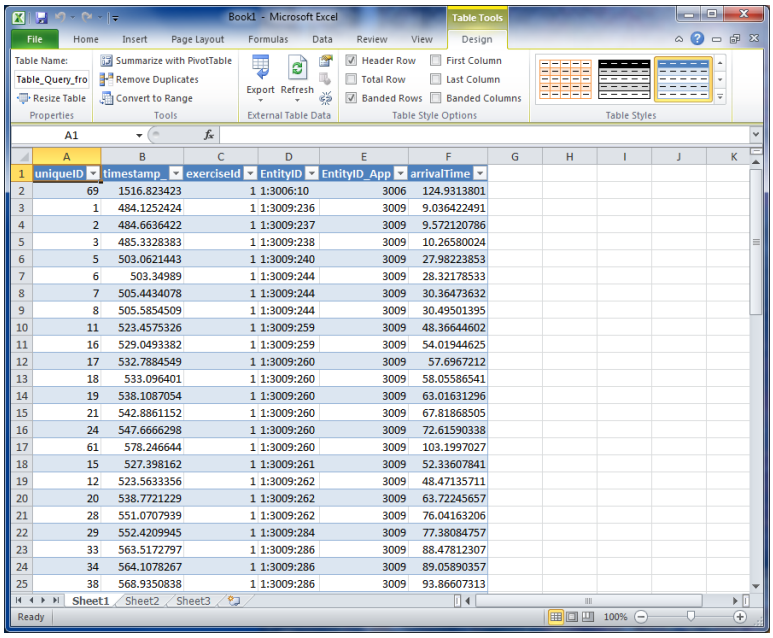


Figure 13-19. Spreadsheet with data imported

13.7.2. 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 13-11.
2. Export a Logger file to Access, as described in "Exporting Data to a Database," on page 13-5.
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 13-20).

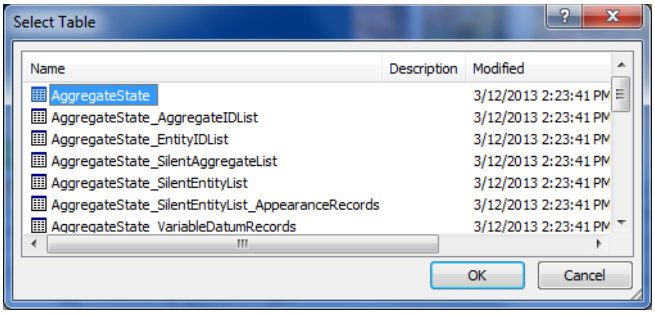


Figure 13-20. Select Table dialog box

8. Select the table that you want to import.
9. Click OK. The data is imported (Figure 13-21).

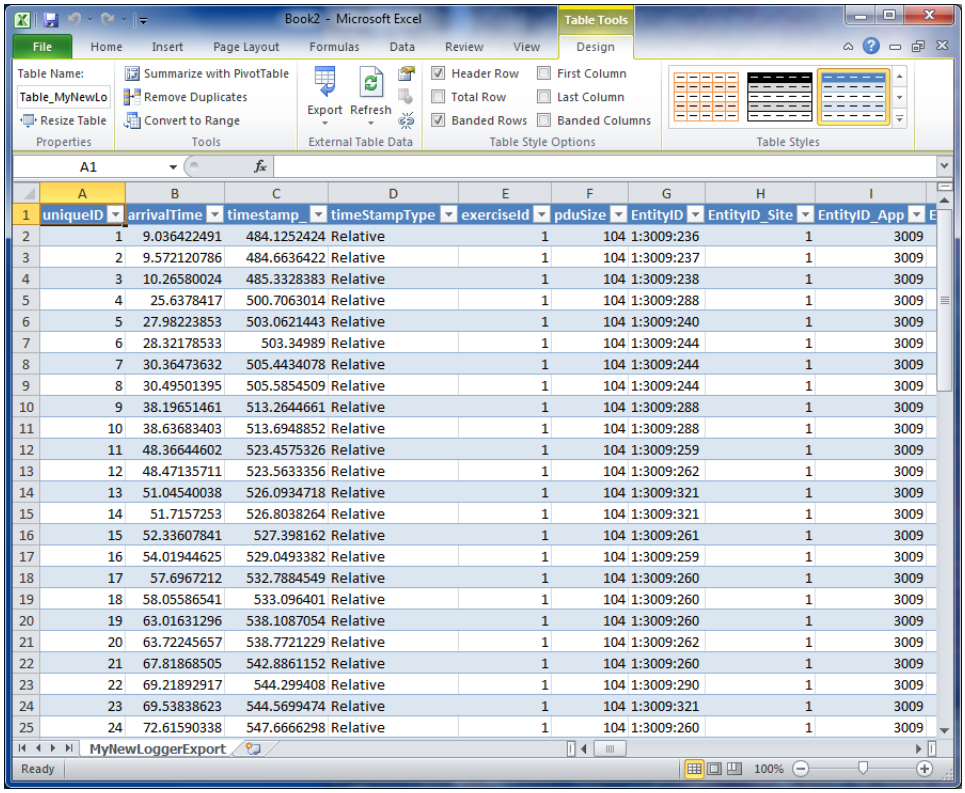


Figure 13-21. Logger file data imported from Access to Excel

14. 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 14-2
Using Logger Commands in the Configuration File 14-3
Using Logger Commands in the Text Interface 14-3
- Entering Commands 14-4
- Displaying a List of Commands 14-5

Command Descriptions 14-5
- Argument Types 14-6
- System Domain Commands 14-7
- Player Domain Commands 14-8
- Recorder Domain Commands 14-10
- Simulation Domain Commands 14-12
- Editor Domain Commands 14-20

Command List 14-21

14.1. Using Logger Commands on the Command Line

When you start the Logger from the Windows or Linux command line, 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.

If a command has multiple parameters, you must specify values for all of them, even if you only want to set some of them. Also, the parameter names used in the command descriptions are not keywords. They are just mnemonics. The commands must use the type of value specified, such as a boolean value or an integer. For example, `SetRemoteControlSettings` is described as follows:

`SetRemoteControlSettings` *`useSimChannel`*(bool) *`useBackChannel`*(bool) *`record`*(bool) *`allowDelete`*(bool) *`allowAll`*(bool) *`sendResponses`*(bool) *`crippleGui`*(bool) *`useLegacyPdu`*(bool) *`connectSendInterval`*(double) *`heartbeatSendInterval`*(double)

To use `SetRemoteControlSettings` in a command, you would have to do something like this (for DIS):

```
loggerDIS.exe -r "SetRemoteControlSettings true true true
  false false true false true 5.0 5.0"
```

Or this (for HLA):

```
loggerHLA13.exe -r "SetRemoteControlSettings true true true
  false false true false true 5.0 5.0"
```

This example starts the Logger and opens the file *../logger_tapes/FollowSuspect-2019-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.)

```
logger<version> -r "OpenFile \"../logger_tapes/FollowSus-
  pect-2019-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:

  ```
  logger<version> "../logger_tapes/FollowSus-
  pect-2019-DIS.lgr"
  ```

- If a command specifies the name of a recording, and there is a video file with the same name, it gets loaded.

This 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.Physi-
  calEntity.Platform.Aircraft ObjectRoot.BaseEntity.Physi-
  calEntity.Radio InteractionRoot.WeaponFire\""
```

14.2. 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 5-23.

14.3. 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 appropriate *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.

14.3.1. 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 14-5.) 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.PhysicalEn-
  tity.Platform.Aircraft" "ObjectRoot.BaseEntity.Physical-
  Entity.Radio" "InteractionRoot.WeaponFire"
```

You can manipulate the command line as follows:

- To execute a command, press **ENTER**.
- To reset a command entry, press **ESC**.
- 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.

14.3.2. 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**.

14.4. Command Descriptions

Commands are organized into domains. If you add a command to *lgrConfig.xml*, you must specify its 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

14.4.1. 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 14-1 describes the argument types used for commands.

Table 14-1: Argument types

Argument Type	Description
int	32-bit signed integer. For example: `SetNotifyLevel 2`
int64	64-bit signed integer. For example: `SetStepSize 1000000`
double	64-bit floating point number. For example: `SetTimeoutRate 5.0`
stringlist	A series of strings, separated by spaces. The individual strings cannot have spaces. The entire series is surrounded by double quotes. For example: `"string1 string2 string3"`
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).

14.4.2. System Domain Commands

Table 14-2 describes the commands in the System domain.

Table 14-2: System domain commands

Command	Description
`DeleteFile` *`filename`*(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` *`pluginName`*(string)	Loads the specified plug-in.
`SetDefaultTapeDirectory` *`defaultTapeDir`*(string)	Sets the directory where the Logger creates and loads files.
`SetEntityTracking` *`entityTracking`*(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` *`fastJumpsWithoutDiff`*(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.
`SetGuiConfigFilename` *`filename`*(string)	Specifies the name of the GUI layout configuration file. Default: *logger-gui.conf*.
`SetNotifyLevel` *`vlLevel`*(int) *`lgrLevel`*(int)	Sets the notification level of VR-Link and the Logger.
`SetPlaybackReadTimeLimit` *`playbackReadTimeLimit`*(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` *`directory`*(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 *temp* 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.

14.4.3. Player Domain Commands

Table 14-3 describes the commands in the Player domain.

Table 14-3: Player domain commands (Sheet 1 of 2)

Command	Description
`ClosePlaybackFile`	Closes the current playback file.
`OpenFile` *`filename`*(string) *`constructFilename`* (bool)	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.
`Pause`	Pauses playback. This command has no effect if the Logger does not have a file open or is already paused.
`Play`	Starts playback. If the Logger is already playing this command has no effect.
`PlayPause`	Toggles between the paused and play state. If the Logger is stopped, it starts playback.
`Restart`	Starts playing from the beginning of the file.
`RevertFile`	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.
`SaveFile` *`reopenAfterSave`*(bool)	Saves a previously saved, but currently unsaved file to its current file name. The previous file is backed up to a file named *file-name.bak.lgr*, for example: *test.lgr* would get saved to *test.bak.lgr*. The current file contains the latest changes. Saving the file closes it. The reopenAfterSave boolean specifies if the saved file should be reopened after the save.
`SaveFileAs` *`filename`*(string) [*`description`*(string) *`writeDescription`*(bool) *`reopenAfterSave`*(bool)]	Saves the current file to a new file. Saving the file closes it. The reopenAfterSave boolean specifies if the saved file should be reopened after the save. Optionally, add a description.
`SetBackupOnSave` *`backup`*(bool)	Enables or disables automatic backup when you save a previously saved file.
`SetCurrentTime` *`time`*(time)	Moves the timeline cursor to the specified time.

Table 14-3: Player domain commands (Sheet 2 of 2)

Command	Description
`SetFollowingInterval` *`interval`*	Specifies the default length of the zoom timeline, in seconds.
`SetFollowingExportInterval` *`interval`*	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.
`SetLoopAtEnd` *`value`*(bool)	Enables or disables looping to start at the end of playback.
`SetPacketViewInterval` *`interval`* (double)	Specifies the interval between tick marks in the Packet Count View and Packet Size View.
`SetPlaybackSpeed` *`speed`*(double) *`skippingPlayback`* (bool)	Sets the playback speed of the Logger. If *`skippingPlayback`* is True, skip from checkpoint to checkpoint without sending the interim state updates.
`SetPlayOnLoad` *`value`*(bool)	Enables starting playback immediately after opening a file.
`SetSkippingPlaybackThreshold` *`skippingPlaybackThreshold`*(double)	Set the playback speed above which skipping playback mode is used automatically.
`SetStartStopTimes` *`startTime`*(time) *`stopTime`*(time)	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.
`SkipTime` *`time`*(double) *`forward`*(bool)	Moves the timeline cursor the specified time (forward or backward) relative to the current time.
`StopPlayback`	Stops playback. This command has no effect if the Logger is not in the playback state.
`Unpause`	Unpauses the Logger. This command has no effect if the Logger is not paused.

14.4.4. Recorder Domain Commands

Table 14-4 describes the commands in the Recorder domain.

Table 14-4: 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` *`filename`*(string) *`description`*(string) *`constructFilePathAndName`*(bool) *`overwrite`*(bool)	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 `newFile`.
`RecordToMultipleFiles` *`fileName`*(string)	Starts recording to new files.
`SetCheckpointParameters` *`policy`*(int) *`size`*(int) *`count`*(int) *`time`*(time)	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` *`record`*(bool)	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` *`duration`*(double) *`namingConvention`*(string)	Set the multiple file recording parameters.

Table 14-4: Recorder domain commands (Sheet 2 of 2)

Command	Description
`SetStopCondition` *`condition`*(int) *`close`* (bool)	Specifies the stop condition for recording as follows: ◆ 0 — no condition. ◆ 1— duration (enables `SetStopDuration`). ◆ 2 — filesize (enables `SetStopSize`). If the *`close`* parameter is True, the file is closed when recording is stopped.
`SetStopDuration` *`duration`*(time)	Stops recording at the specified time if `SetStopCondition` is set to 1.
`SetStopSize` *`filesize`*(int64)	Stops recording when the file reaches *`filesize`* size if `SetStopCondition` is set to 2.
`StopRecording` *`closeAndOpenForPlayback`*(bool)	Stops recording and ends. By default *`closeAndOpenForPlayback`* is true (1). If you set *`closeAndOpenForPlayback`* to false (0), then after you issue `StopRecording`, you are still in record mode. You would need to issue the `CloseRecordFile` command to close the file and exit record mode.
`UnpauseRecording`	Resume a paused recording.

14.4.5. Simulation Domain Commands

Table 14-5 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 14-5: Simulation domain commands (Sheet 1 of 8)

Command	Description
Protocol-independent Commands	
`Connect`	Connects to the network simulation using the previously set parameters.
`Disconnect`	Disconnects from the network simulation.
`FilterEntityTypes` *entityType1* ... *entityTypen*(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` *address*(string) *port*(int) *interface*(string)	Set the parameters for the back channel connection configuration.
`SetCommonEntityTypes` *entityType1* ... *entityTypen*(string)	Specifies a list of entity types to list in the Filter by Entity Type dialog box. Does not enable filtering of these entity types.
`SetListenWhenIdle` *state* (bool)	Enables or disables listening to the network when the Logger is neither recording or playing a file.
`SetRemoteControlId` *remoteControlId* (integer)	Specifies the Logger's remote control ID. By default the value is assigned the Logger's site ID. This value is used if `SetRemoteControlSettings allowAllRemoteControl` is disabled.

Table 14-5: Simulation domain commands (Sheet 2 of 8)

Command	Description
SetRemoteControlSettings *useSimChannel*(bool) *useBackChannel*(bool) *record*(bool) *allowDelete*(bool) *allowAll*(bool) *sendResponses*(bool) *crippleGui*(bool) *useLegacyPdu*(bool) *connectSendInterval*(double) *heartbeatSendInterval*(double)	Specifies how the Logger responds to Logger Control messages. ◆ *useSimChannel* determines whether the Logger responds to Logger control messages from the exercise connection. ◆ *useBackChannel* determines whether the Logger responds to Logger control messages from the Remote Control Logger. ◆ *record* determines whether or not Logger control messages are included in the file. ◆ *allowDelete* determines whether or not to respond to the delete file Logger control message. ◆ *allowAll* determines whether remote control PDUs are filtered based on the Logger's site-application-enityNum triplet, where the Logger's remote control ID is used for the enityNum. 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. ◆ *sendResponses* 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). ◆ *crippleGui* 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. ◆ *useLegacyPdu* 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. ◆ *connectSendInterval* 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 14-5: Simulation domain commands (Sheet 3 of 8)

Command	Description
SetRemote-ControlSet-tings (continued)	◆ *heartbeatSendInterval* 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 *heartbeatSend-Interval* is less than 0, then the heartbeat message is not sent.
SetSaveConnectionConfiguration *saveConnectionConfiguration* (bool)	Enables or disables saving of connection information to the */grConfig.xml* file.
SetScaleAttributes *scale*(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 *state* (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 *filter*(bool) *loId*(int) *hiId*(int)	Configures VR-Forces message filtering. The filter argument enables or disables filtering of VR-Forces messages. The *loId* argument specifies the lowest message ID to filter out. The *hiId* argument specifies the highest message ID to filter out.
HLA-specific Commands	
CommonObjectNames *objectName1 ... objectNameN*(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 *state* (bool)	Enable or disable whether request updates is invoked when an object is discovered. Default: true.
FilterObjectNames *objectName1* ... *objectNameN*(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 *filename*(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 14-5: Simulation domain commands (Sheet 4 of 8)

Command	Description
SaveSubscriptionListFile *filename*(string)	Saves the subscription settings to an XML file with the specified name. In the GUI, the extension *.subx* is added to filenames, but it is not added to the filename when saved from the command line.
SetConnectionParametersHLA *federateName*(string) *federationName*(string) *fedFile*(string) *ignoreAdvisories*(bool) *fomVersion*(double) *fomRevision*(int) *useMatrexFomMapper*(bool) *matrexFomMapperData*(string) *fomMapper*(string) *fomMapperData*(base64) *useEmptyFomMapper*(bool) *noRtiCompilerCheck*(bool) SetConnectionParametersHLAe *federateName*(string) *federateType*(string) *federationName*(string) *fedFile*(string) *fomModules*(string) *MIM*(string) *ignoreAdvisories*(bool) *fomVersion*(double) *fomRevision*(int) *useMatrexFomMapper*(bool) *matrexFomMapperData*(string) *fomMapper*(string) *fomMapperData*(base64) *useEmptyFomMapper*(bool) *noRtiCompilerCheck*(bool)	Specifies HLA (1.3 and 1516) or HLA Evolved connection parameters: ◆ *federateName*. The name of the Logger federate. (HLA Evolved only) ◆ *federateType*. The type of the Logger federate. ◆ *federationName*. The name of the federation execution. ◆ *fedFile*. The name of the FED file (or FDD or XML file). ◆ *fomModules*. One or more FOM module files. (HLA Evolved only) ◆ *MIM*. A MIM file in the *bin64* folder. (HLA Evolved only) ◆ *ignoreAdvisories*. Configures the Logger to ignore RTI advisory queues and always send data. Default: False. ◆ *fomVersion*. The RPR FOM version to use. ◆ *fomRevision*. The RPR FOM revision to use. ◆ *useMatrexFomMapper*. Tells the Logger whether to use the appropriate Matrex FOM Mapper. (Vversions for HLA 1.3, HLA 1516, and HLA Evolved are in the Logger's *bin64* folder. Default: False. ◆ *matrexFomMapperData*. Initialization data for the Matrex FOM Mapper — version = 1.0, for example. ◆ *fomMapper*. The name of the FOM Mapper to use, if any. ◆ *fomMapperData*. Initialization data for the FOM Mapper. ◆ *useEmptyFomMapper*. The Logger uses 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. ◆ *noRtiCompilerCheck*. Controls whether the Logger checks the compiler version of the RTI libraries on initialization. The Logger exits if it finds mismatched versions. Set to True to skip the compiler check. Default: False.

Table 14-5: Simulation domain commands (Sheet 5 of 8)

Command	Description
`SetDdmParameters` *`enableDdm`*(bool) *`playbackMode`*(string) *`subscribeMode`*(string) *`playbackDynamicAssociation`*(bool) *`playbackDynamicExtents`*(bool) *`allowPlaybackWithoutRegions`*(bool) *`recordConveyedRegions`*(bool) *`requestUpdates`*(bool) *`subscribeUnspecifiedAttributes`*(bool)	Specifies DDM parameters, 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 plug-in extension. – 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.

Table 14-5: Simulation domain commands (Sheet 6 of 8)

Command	Description
	◆ *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. ◆ *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.
`SetDisableTimeManagementTest` *disableTesting*(bool)	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 *disableTesting* to true turns off the test. However, time management is disabled.
`SetDrainInput` *maxTime*(double), *minTime*(double)	Specifies the minimum and maximum amount of time, in seconds, to spend reading messages from the exercise connection (that is, the network) each tick.
`SetDynamicNamePrefix` *prefix*(string)	Specifies the prefix that the Logger attaches to the beginning of object names that begin with HLA_ when it is recording exercises.
`SetHeartbeatIntervals` "*className*(string):*interval*(float):*inheritable*(bool) ..."	Specifies a heartbeat interval for playback. You can specify multiple classes. For details, please see "Configuring HLA Heart-beating," on page 5-8.
`SetIgnoreAdvisories` *mode*(bool)	Specifies the Logger's response to HLA advisory messages. You must specify this command before the `Connect` command. Default: false.

Table 14-5: Simulation domain commands (Sheet 7 of 8)

Command	Description
`SetPassiveSubscription` *`mode`*(bool)	Configures passive subscription for HLA. You must specify this command before the `Connect` command. Default: false.
`SetSubscriptionLists` *`objectClasses`*(stringList) *`interactionClasses`*(stringList)	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.
`SetTimeManagement` *`timeConstrainedRecording`*(bool) *`timeConstrainedPlayback`*(bool) *`timeRegulatingRecording`*(bool) *`timeRegulatingPlayback`*(bool) *`lookahead`*(double)	Configures time management options. The *`timeConstrainedPlayback`* and *`timeRegulatingRecording`* arguments must be included on the command line, but are not supported and are ignored.
`SetTimeoutIntervals` "*`className`*(string):*`interval`*(float):*`inheritable`*(bool) ..."	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	
`AddCustomPDU` *`pduName`*(string) *`pduKind`*(string) *`entityIdOffset`*(string) *`entityTypeOffset`*(string)	Specifies a PDU to add to the Filter by PDU Kind dialog box.
`CommonEntityIds` *`id1 ... idN`*(string)	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box. Does not enable filtering of these entity IDs.
`FilterEntityIds` *`id1 ... idN`*(string)	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box and enables filtering for these entity IDs.

Table 14-5: Simulation domain commands (Sheet 8 of 8)

Command	Description
`SetConnectionParametersDIS` *`port`*(int) *`exerciseId`*(int) *`siteId`*(int) *`appNumber`*(int) *`useAsynchIO`*(bool) *`sendBufferSize`*(int) *`recvBufferSize`*(int) *`multicastTtl`*(int) *`destAddr`*(string) *`subnetMask`*(string) *`deviceAddr`*(string) *`mcastAddresses`*(stringList)	Specifies parameters for the DIS exercise connection, as follows: ◆ *`port`*. The port to listen to. ◆ *`exerciseId`*. The Logger's exercise ID. ◆ *`siteId`*. The Logger's site ID. ◆ *`appNumber`*. The Logger's application number. ◆ *`useAsynchIO`*. Enable (1) or disable (0) asynchronous I/O. ◆ *`sendBufferSize`*. The size of the send buffer. ◆ *`recvBufferSize`*. The size of the receive buffer. ◆ *`multicastTtl`*. The multicast time-to-live, in seconds. ◆ *`destAddr`*. The destination address to send to. ◆ *`subnetMask`*. The subnet mask used for determining the destination when *`destAddr`* is not unspecified. ◆ *`deviceAddr`*. The network interface address to use. ◆ *`mcastAddresses`*. A list of multicast addresses, separated by spaces.
`SetDrainInput` *`maxTime`*(double), *`numPdusToRead`*(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` *`rate`*(double)	Specifies the DIS heartbeat rate.
`SetTimeoutRate` *`rate`*(double)	Specifies the DIS timeout rate.

14.4.6. Editor Domain Commands

Table 14-6 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 14-6: Editor domain commands (Sheet 1 of 2)

Command	Description
`AppendFileToEnd` *`filename`*(string)	
	Appends the file *`filename`* to the end of the open file.
`ClearSection` *`start`*(time) *`end`*(time)	
	Removes all packets between the *`start`* and *`end`* times in a file.
`ConvertFile`	Converts the current file to the latest file format. This command is not valid for files that are already in the latest file format.
`CutSection` *`start`*(time) *`end`*(time)	
	Removes all packets between the *`start`* and *`end`* times in a file. All packets after the *`end`* time are offset so that the first packet after the *`end`* time is now at the *`start`* time.
`ExportFile` *`exportFormat`*(string) *`destination`*(string) *`checkpoint`*(bool) *`openAfterExport`*(bool)	
	Exports a file using a given *`exportFormat`* to a *`destination`*. The last two parameters are only used when exporting a Logger file. The Logger tries to export using the *`exportFormat`* and the *`destination`*. If *`checkpoint`* is true and the file is a Logger file, new checkpoints are generated for the output. If *`openAfterExport`* is true and the file is a Logger file, the Logger closes the current file and tries to open the new file. The valid *`exportFormat`* 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 *`destination`* typically is a filename (but does not need to be) and is specific to the *`exportFormat`*. The Plain Text exporter exports to any filename, but is typically exported to a *.txt* file. The Logger File exporter exports to a *.lgr* file and adds that extension if it is not in the *`destination`* string. Other exporters may encode the *`destination`* differently.

Table 14-6: Editor domain commands (Sheet 2 of 2)

Command	Description
`ExportSection` *`exportFormat`*(string) *`destination`*(string) *`start`*(time) *`end`*(time) *`offset`*(double) *`checkpoint`*(bool) *`openAfterExport`*(bool)	Exports a section of a file using a given *`exportFormat`* to a *`destination`*. This command works just like the `ExportFile` command except that it exports a segment of a file rather than the entire file.
`MergeInFile` *`filename`*(string) *`offset`*(double)	Merges the Logger file *`filename`* into the open file. An optional offset is applied to all packets from the file before being combined with the current data.
`OffsetTime` *`offset`*(double)	Offsets the time of the packets in the open file by the *`offset`* value. The *`offset`* 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.
`PrefixFileToStart` *`filename`*(string)	Inserts the file named *`filename`* at the start of the open file.
`TrimStart`	Removes any dead time from the start of the file and offsets all packets so that the first packet time stamp is 0.0.

14.5. Command List

Table 14-7 section lists all Logger commands and their domain:

Table 14-7: Logger commands

Command	Domain
AddCustomPDU	Simulation
AppendFileToEnd	Editor
CancelRecording	Recorder
ClearSection	Editor
ClosePlaybackFile	Player
CloseRecordFile	Recorder
CommonEntityIds	Simulation
CommonObjectNames	Simulation
Connect	Simulation
ConvertFile	Editor
DeleteFile	System

Table 14-7: 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
SetConnectionParametersDIS	Simulation
SetConnectionParametersHLA	Simulation
SetConnectionParametersHLAe	Simulation

Table 14-7: Logger commands

Command	Domain
SetCurrentTime	Player
SetDdmParameters	Simulation
SetDefaultTapeDirectory	System
SetDisableTimeManagementTest	Simulation
SetDrainInput	Simulation
SetDynamicNamePrefix	Simulation
SetEntityTracking	System
SetFastJumpsWithoutDiff	System
SetFollowingInterval	Player
SetFollowingExportInterval	Player
SetGuiConfigFilename	System
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

Table 14-7: Logger commands

Command	Domain
SetSubscriptionLists	Simulation
SetTempDir	System
SetTimeoutRate	Simulation
SetVRFFiltering	Simulation
SkipTime	Player
StopPlayback	Player
StopRecording	Recorder
TrimStart	Editor
Unpause	Player
UnpauseRecording	Recorder

15. 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 15-2
Configuring Loggers to be Controlled Remotely 15-2
Starting the Logger Remote Control 15-3
Recording Exercises with the Logger Remote Control 15-3
Playing Exercises with the Logger Remote Control 15-5
Configuring the Logger Remote Control Connection 15-6
Specifying the Refresh Rate 15-6
Setting the Timebar Color 15-7
Exiting the Logger Remote Control and Connected Loggers 15-7
Logger Remote Control Command-line Options 15-8

15.1. 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 — *./bin64/lgrControlGui*. It is protocol-independent. It can control DIS Loggers, HLA Loggers, or both.

15.2. Configuring Loggers to be Controlled Remotely

By default, Loggers are configured to be controlled using the Logger Remote Control. It can accept remote control messages from the exercise connection (**useSimChannel**) and from the back channel (**useBackChannel**). You can configure it to accept remote control messages from both sources, only one of them, or neither of them.

To enable remote control set the following variables to true, as shown:

```
<var name="useSimChannel" 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 Loggers 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.

15.3. Starting the Logger Remote Control

➤ To start the Logger Remote Control on Windows, on the Start Menu, choose **MÄK Technologies → Logger Remote Control**. The Logger Remote Control opens (Figure 15-1).

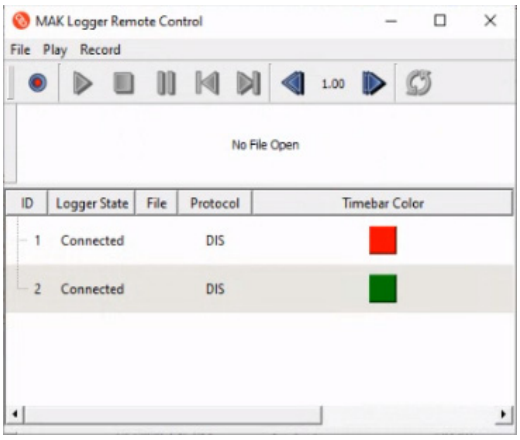


Figure 15-1. Logger Remote Control

To start the remote Logger GUI on Linux or from the Windows command line:

1. Change directory to *./bin64*.
2. Run *lgrControlGui*.

15.4. 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 *./logger-tapes/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.

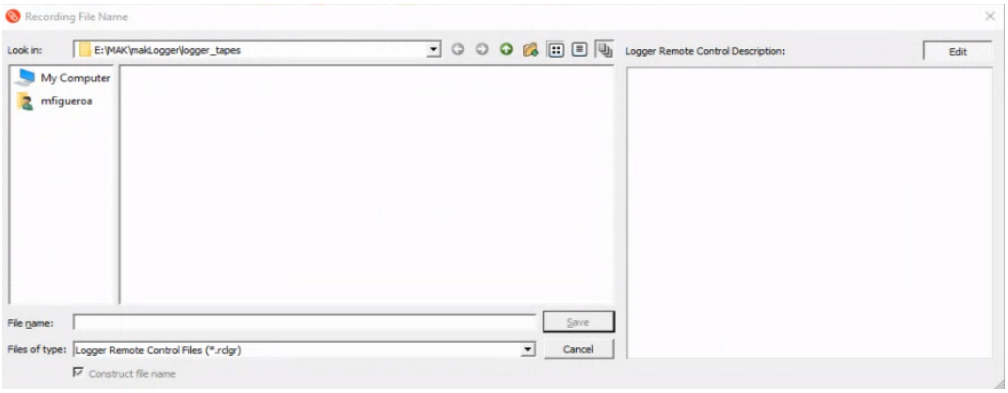


Figure 15-2. Recording File Name dialog box

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 path-name, 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.

15.5. 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.

15.6. 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 15-3).

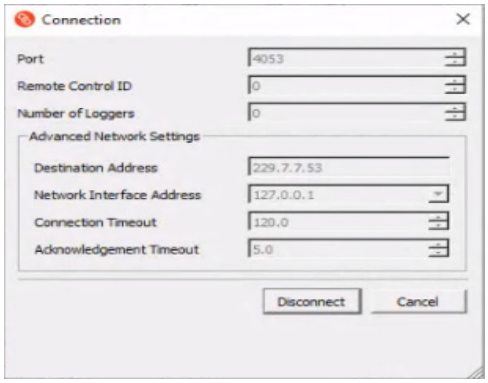


Figure 15-3. Connection dialog box

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.

15.7. 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.

15.8. 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 15-4). This will alert you to the possibility that there is a problem with one of the Loggers.

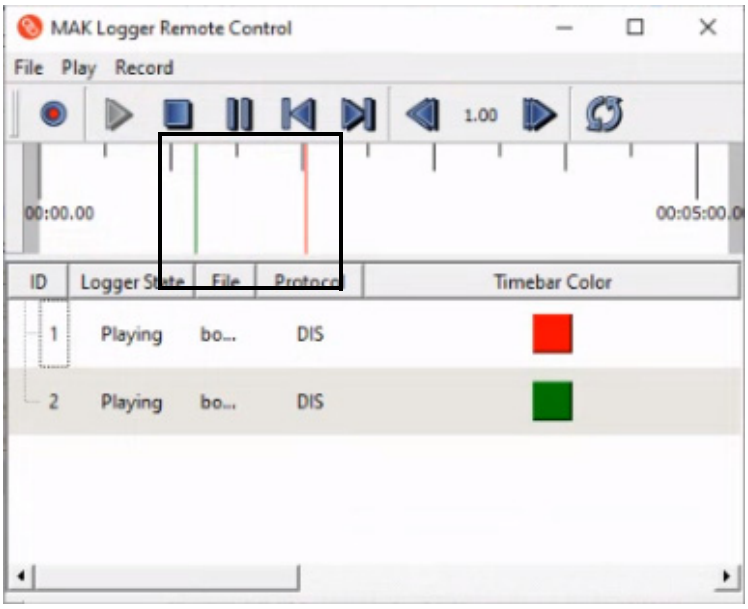


Figure 15-4. Logger Remote Control timebars

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.

15.9. 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**.

15.10. Logger Remote Control Command-line Options

You can run the Logger Remote Control from the command line. Table 15-1 lists command-line options.

Table 15-1: Logger Remote Control command-line options

Option	Description
`(-- | --ignore_rest)`	Ignores the rest of the labeled arguments following this flag.
`--acknowledgeTimeout` *timeout*	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 acknowledgment is not received within this timeout interval, the command is considered lost.
`--backChannelAddress` *address*	The destination address for back channel communication.
`--backChannelInterface` *address*	The interface to use for the back channel communication.
`--backChannelPort` *port*	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` *timeout*	The amount of time, in seconds, to wait before timing out a connection. Default: 120 seconds.
`(-h | --help)`	Displays usage information and exits.
`--numberOfLoggers` *number*	The maximum number of Loggers to connect to. 0 means connect to all available Loggers.
`--remoteControlId` *ID*	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:

   `logger`*`version`* `-A` *`address`* `-P` *`port`*

   where address and portnum are the broadcast address and UDP port used by your other applications, for example:

   `logger`*`version`* `-A 123.456.999.255 -P 3020`

B. FED File Naming Conventions

This appendix describes the naming conventions used for FED files.

FED File Naming Convention B-2

B.1. FED File Naming Convention

Logger 5.9 includes FED files to support FOM mappers for different versions of the HLA specification. Logger 5.4 and later use the FOM naming conventions introduced in VR-Link 5.2. Prior releases used different names for FED files. This section describes the newer conventions and how to map older FED files to the new ones.

Historically, VR-Link named FED files using the convention VR-Link<*RPR FOM version*>.fed or VR-Link<RPR FOM version>.xml. For example, the FED file for RPR FOM 2, draft 17 was *VR-Link20017-1.fed*. This naming scheme had two problems. First, it was confusing for end-users of applications that use VR-Link (but who are not themselves VR-Link customers) to be using a FOM named that way. Second, the filename was not versioned, so any updates that we would make to the existing FOMs could cause compatibility problems. In fact for this latter reason, we had refrained from fixing a few minor bugs that we found over the years.

Effective with VR-Link 5.2, the naming convention is: MAK-<*FOM name*><*FOM revision*><FOM *version*>.fed (or *.xml*, or *_evolved.xml*, as appropriate). For example, *VR-Link20017-1.fed* was renamed *MAK-RPR20017-1-1.fed*.

Additionally, given the power of HLA Evolved FOM Modules, starting with RPR 2.0 we provide the RPR standard FOM for HLA Evolved, currently named RPR_FOM_v2.0_1516-2010. MAK's extensions are in FOM Modules that our software will automatically add to an exercise. This means that if you have another RPR-based FOM, such as NETN or NASMP, you will have a much easier time using it without needing to modify the FOM.

Table B-1 lists legacy FED files and how they map to new FED files. Many of the FED files delivered in previous releases do not have updated versions, usually because they are rarely used.

Table B-1: Old FED file to new FED file mappings

Old FOM	New FOM
HLA 1.3	
DI-Guy-13_0.fed	
DI-Guy.fed	
VR-Link.fed	MAK-RPR1-1-1.fed
VR-Link05.fed	
VR-Link07.fed	
VR-Link08.fed	
VR-Link10.fed	MAK-RPR1-1-1.fed
VR-Link16.fed	
VR-Link20006.fed	

Table B-1: Old FED file to new FED file mappings

Old FOM	New FOM
VR-Link20014.fed	
VR-Link20017-1.fed	MAK-RPR20017-1-1.fed
VR-Link20017-6.fed	MAK-RPR20017-6-1.fed
VR-Link20017.fed	
VR-Link20020.fed	MAK-RPR2-1-1.fed
VR-LinkBOML16.fed	
VR-LinkDDM.fed	
VR-LinkMC02.fed	MAK-RPR1-MC02-1-1.fed
VR-LinkTM.fed	MAK-RPR1-TimeManagement-1-1.fed
VRL-20017-1-Link16.fed	
VRL-DIGuy20017.fed	
VRL-DIGuy20017DDM.fed	
1516-2000	
DI-Guy-13_0.xml	
DI-Guy.xml	
VR-Link.xml	MAK-RPR1-1-1.xml
VR-Link20014.xml	
VR-Link20017-1.xml	MAK-RPR20017-1-1.xml
VR-Link20017-6.xml	MAK-RPR20017-6-1.xml
VR-Link20020.xml	MAK-RPR2-1-1.xml
VR-LinkBOML16.xml	
VR-LinkDDM.xml	
VR-LinkTM.xml	MAK-RPR1-TimeManagement-1-1.xml
VRL-DIGuy20017.xml	
VRL-DIGuy20017DDM.xml	
HLA Evolved	
DIGuyModule_1300.xml	MAK-DIGuy-1_evolved.xml
DIGuyModule_1302.xml	
RPR_FOM_v2.0_draft20_1516-2010.xml	RPR_FOM_v2.0_1516-2010.xml
VR-Link20014_base_evolved.xml	
VR-Link20014_evolved.xml	
VR-Link20017-1_base_evolved.xml	MAK-RPR20017-Base-1-1_evolved.xml

Table B-1: Old FED file to new FED file mappings

Old FOM	New FOM
VR-Link20017-1_base_evolved_with-UpdateRate.xml	
VR-Link20017-1_evolved.xml	MAK-RPR20017-1-1_evolved.xml
VR-Link20017-1_evolved_withUpdat-eRate.xml	
VR-Link20017-6_base_evolved.xml	MAK-RPR20017-Base-6-1_evolved.xml
VR-Link20017-6_evolved.xml	MAK-RPR20017-6-1_evolved.xml
VR-LinkDDM_base_evolved.xml	
VR-LinkDDM_evolved.xml	
VR-Link_LgrControl_evolved.xml	MAK-LgrControl-1_evolved.xml
VR-Link_ViewControl_evolved.xml	
VR-Link_VrlExt_evolved.xml	MAK-VRFExt-1_evolved.xml
VR-Link_base_evolved.xml	MAK-RPR1-Base-1-1_evolved.xml
VR-Link_evolved.xml	MAK-RPR1-1-1_evolved.xml
VRL-DIGuy20017DDM_evolved.xml	
VRL-DIGuy20017_evolved.xml	
updateRate.xml	MAK-updateRate-1_evolved.xml
VR-LinkBOML16_module.xml	MAK-BOML16-1_evolved.xml
	MAK-VRFAggregate-1_evolved.xml

MAK Products Glossary

This glossary defines terms used in MAK product documentation.

absolute dead reckoning

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

aggregate

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

API

Application Programming Interface.

articulated part

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

attached part

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

body coordinate frame

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

channel

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

clipping

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

clipping planes

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

culling

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

Cartesian coordinates

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

Data Distribution Management

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

dead-reckoning

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

DDM

Data Distribution Management.

Defense Modeling and Simulation Office

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

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

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

Distributed Interactive Simulation

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

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

DMSO

See Defense Modeling and Simulation Office.

element definition

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

emitter

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

entity

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

entity maintenance

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

entity-type

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

The components of an entity-type are:

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

environmental process

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

Euler angles

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

event

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

exercise

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

exercise connection

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

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

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

federate

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

federation

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

Federation Object Model

Defines the data content of a federation execution.

federation execution

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

field of view

Controls the perspective of the observer.

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

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

filter range

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

FOM

See Federation Object Model.

FOM Module

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

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

frame rate

The rate at which the application displays updated images.

ground clamping

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

geocentric coordinates

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

geodetic coordinates

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

gridded data

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

guise

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

head-up display

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

heartbeat

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

High-Level Architecture

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

HLA

See High-Level Architecture.

interaction

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

Logger control PDUs

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

logical range

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

Local RTI Component

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

LRC

See Local RTI Component.

MAK Technologies Lisp

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

message

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

MIM

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

model definition

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

Modeling and Simulation Coordination Office

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

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MAK Technologies Lisp.

orientation clamping

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

object

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

object handle

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

object name

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

observer

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

PDU

See Protocol Data Unit.

Protocol Data Unit

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

radio transmitter

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

radio receiver

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

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

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

reflected entity list

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

reference FOM

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

relative dead reckoning

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

remote display engine

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

RID file

See RTI Initialization Data.

RTI Initialization Data

The initialization data required by the RTI for operation.

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

RPR FOM

See Realtime Platform Reference FOM.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

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

scene element

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

Simulation Interoperability Standards Organization

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

simulation time

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

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

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

state

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

tape

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

terrain following

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

timeout

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

timescale

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

topographic coordinates

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

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

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

UDP port

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

Universal Transverse Mercator

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

UTM

See Universal Transverse Mercator.

view control messages

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

Index

-- command-line option 5-28
--appNumber command-line option 4-21, 5-30
--configFile command-line option 5-23, 5-28
--defaultTapeDir command-line option 5-28
--destAddrString command-line option 4-21, 5-29
--deviceAddrString command-line option 4-23, 5-30
--disableTMTesting command-line option 5-29
--disPort command-line option 4-20, 5-30
--disVersion command-line option 5-30
--enableTrafficAnalyzer command-line option 5-29
--execName command-line option 4-8, 5-29
--exerciseId command-line option 4-20, 5-30
--federateName command-line option 4-7, 5-29
--federateType command-line option 4-7, 5-29
--fedFileName command-line option 4-8
--filterEntityIds command-line option 5-30
--filterEntityTypes command-line option 5-28
--filterObjectNames command-line option 5-29
--fomMapperData command-line option 4-13, 5-29
--fomMapperLib command-line option 4-11, 5-29
--fomModules command-line option 4-9, 5-29
--ignore_rest command-line option 5-28
--ignoreAdvisories command-line option 4-11, 5-29
--logFileName command-line option 5-28
--maxDrainInputTime command-line option 5-29
--mcastAddresses command-line option 4-26, 5-30
--mcastTtl command-line option 4-23, 5-30
--messageLevel command-line option 5-28
--MIM command-line option 4-9, 5-29
--notifyLevel command-line option 5-28
--receiveBufferSize command-line option 4-22
--recvBufferSize command-line option 5-30
--rprFomRevision command-line option 4-10, 5-29
--rprFomVersion command-line option 4-10, 5-29
--runCommand command-line option 5-28
--sendBufferSize command-line option 4-22, 5-30
--showConsole command-line option 5-28
--siteId command-line option 4-20, 5-30
--timeConstrainingRecording command-line option 5-29
--timeRegulatingPlayback command-line option 5-29
--useEmptyFomMapper command-line option 4-11, 5-29
--version command-line option 5-28
-D command-line option 4-23

A

-A command-line option 4-21, 5-29
-a command-line option 4-21, 5-30
acceleration scaling 3-7
Access
  configuring data source for 13-11
  exporting to 13-10
  importing data to Excel 13-22
AddCustomPDU command 14-18
adding
  annotation 9-2
  command to *lgrConfig.xml* 5-26
  instant annotation 9-3
  variable to *lgrConfig.xml* 5-26
address, multicast 5-30
advisory message
  configuring 14-17
  ignoring 4-11
allowPlaybackWithoutRegions parameter 5-16, 14-16
analyzing, data 11-3
annotating, file 9-2
annotation 9-2
  bounding playback to 9-9
  converting type 9-6
  creating 9-2
  deleting 9-6
  editing 9-5
  importing annotation file 9-5
  instant, adding 9-3
  loading automatically 8-2
  opening annotation file 9-5
  saving file 9-4
  selecting 9-5
  updating 9-4
  zooming timeline to 9-8
AppendFileToEnd command 14-20
appending, file 12-5, 14-20, 14-21
application
  number 4-21, 14-19
    specifying 5-30
argument type 14-6
array 11-6, 11-8
arrayBracketMap parameter 11-13
asynchronous I/O 14-19
  configuring 4-21
attribute
  annotation 9-5
  class 11-5
attribute (continued)
  connection 4-4
  list, hiding 11-3
  object 11-6
  scaling, enabling or disabling 14-14
  updating 11-6
  viewing object 11-5
attribute value update
  requesting 4-18
  sending 4-18
attributeMap parameter 11-13
automatic
  playback 8-2
  recording 7-3

B

backup
  enabling or disabling 14-8
  file 7-9, 14-8
bak file 7-9, 14-8
beginning, moving to 8-11
bracket
  array elements, Traffic Analyzer 11-13
broadcast address 4-21, A-2
buffer, send and receive 4-22
button bar 6-3
Bypass Remote Control File, check box 15-5

C

-c command-line option 5-28
canceling, recording 14-10
CancelRecording command 14-10
changing
  connection settings 4-4
  data delimiter 11-10
  datatype mapping 11-11
  delimiter 11-4
  timebar color 15-7
  traffic analyzer coordinate system 11-12
check box
  Bypass Remote Control File 15-5
  Construct File Name 15-3
checkpoint, skipping to 8-6
checkpoint policy 5-2, 14-10
class
  filtering by (HLA) 10-10
  interaction 11-5

class (continued)
object 11-5
classes, subscribing to 4-15
clearing
file segment 12-3
packets from file 14-20
ClearSection command 14-20
ClosePlaybackFile command 14-8
CloseRecordFile command 14-10
closing
file 6-10, 14-8
recording 14-10
color
timebar, changing 15-7
command 5-2
AddCustomPDU 14-18
adding to *lgrConfig.xml* 5-26
AppendFileToEnd 14-20
CancelRecording 14-10
ClearSection 14-20
ClosePlaybackFile 14-8
CloseRecordFile 14-10
CommonEntityIds 14-18
CommonObjectNames 14-14
Connect 14-12
ConvertFile 14-20
CutSection 14-20
DeleteFile 14-7
Disconnect 14-12
Editor domain 14-20
EnableRequestUpdates 4-18, 14-14
Exit 14-7
ExportFile 14-20
ExportSection 14-21
FilterEntityIds 14-18
FilterEntityTypes 10-4, 14-12
FilterObjectNames 14-14
Help 14-7
HLA simulation domain 14-14
in configuration file 5-25
list of 14-21
LoadPlugin 14-7
LoadSubscriptionListFile 4-17, 5-12, 14-14
MergeInFile 14-21
NewFile 14-10
OffsetTime 14-21
OpenFile 14-8
Pause 14-8
PauseRecording 14-10
command 5-2 (continued)
Play 14-8
Player domain 14-8
PlayPause 14-8
PrefixFileToStart 14-21
Record 14-10
Recorder domain 14-10
RecordToMultipleFiles 14-10
Restart 14-8
RevertFile 14-8
running at startup 5-28
SaveFile 14-8
SaveFileAs 14-8
SaveSubscriptionListFile 4-16, 14-15
SetBackChannelConnection 14-12
SetBackupOnSave 7-9, 14-8
SetCheckpointParameters 5-4, 14-10
SetConnectionParameters 4-6, 4-19, 14-12, 14-15, 14-19
SetCurrentTime 14-8
SetDdmParameters 5-11, 5-13, 5-15, 14-16
SetDefaultTapeDirectory 14-7
SetDisableTimeManagementTest 14-17
SetDrainInput 14-17, 14-19
SetDynamicNamePrefix 3-3, 14-17
SetEntityTracking 5-4, 5-5, 14-7
SetFastJumpsWithoutDiff 5-4, 5-6, 14-7
SetFollowingExportInterval 14-9
SetFollowingInterval 14-9
SetGuiConfigFilename 6-6, 14-7
SetHeartbeatIntervals 5-4, 5-8, 14-17
SetHeartbeatRate 4-24, 14-19
SetIgnoreAdvisories 4-11, 14-17
SetListenWhenIdle 14-12
SetLoopAtEnd 14-9
SetNotifyLevel 5-19, 14-7
SetPacketViewInterval 14-9
SetPassiveSubscription 4-17, 14-18
SetPauseOnRecordAs 14-10
SetPlaybackReadTimeLimit 5-4, 5-7, 14-7
SetPlaybackSpeed 14-9
SetPlayOnLoad 14-9
SetRecordToMultipleFilesSettings 14-10, 1-2
SetRemoteControlId 14-12, 14-13
SetRemoteControlSettings 14-13
SetSaveConnectionConfiguration 14-14
SetScaleAttributes 14-14

command 5-2 (continued)
SetSendOnlyUpdatedValues 4-19, 14-14
SetSkippingPlaybackThreshold 14-9, 1-1
SetStartStopTimes 14-9
SetStopCondition 14-11
SetStopDuration 14-11
SetStopSize 14-11
SetSubscriptionLists 4-16, 5-27, 14-18
SetTempDir 14-7
SetTimeManagement 4-14, 14-18
SetTimeoutIntervals 5-4, 5-9, 14-18
SetTimeoutRate 4-25, 14-19
SetVRFFiltering 14-14
Simulation domain 14-12
SkipTime 14-9
StopPlayback 14-9
StopRecording 14-11
syntax in *lgrConfig.xml* 5-25
System domain 14-7
TrimStart 14-21
Unpause 14-9
UnpauseRecording 14-11
command-line, Logger Remote Control 15-8
command-line interface 14-3
domains 14-5
command-line option 5-2
-- 5-28
-A 4-21, 5-29
-a 4-21, 5-30
--appNumber 4-21, 5-30
-c 5-28
--configFile 5-23, 5-28
-D 4-23, 5-30
--defaultTapeDir 5-28
--destAddrString 4-21, 5-29
--deviceAddrString 4-23, 5-30
--disableTMTesting 5-29
--disPort 4-20, 5-30
--disVersion 5-30
-e 4-20, 5-30
--enableTrafficAnalyzer 5-29
--execName 4-8, 5-29
--exerciseId 4-20, 5-30
-F 4-8, 4-10, 5-29
-f 4-12, 5-29
--federateName 4-7, 5-29
--federateType 4-7, 5-29
command-line option 5-2 (continued)
--fedFileName 4-8
filename 5-28
--filterEntityIds 5-30
--filterEntityTypes 5-28
--filterObjectNames 5-29
--fomMapperData 4-13, 5-29
--fomMapperLib 4-11, 5-29
--fomModules 4-9, 5-29
-h 5-28
--ignore_rest 5-28
--ignoreAdvisories 4-11, 5-29
-l 5-19, 5-28
--logFileName 5-28
-m 5-19, 5-28
--maxDrainInputTime 5-29
--mcastAddresses 4-26, 5-30
--mcastTtl 4-23, 5-30
--messageLevel 5-28
--MIM 4-9, 5-29
-N 4-7, 5-29
-n 5-19, 5-28
--notifyLevel 5-28
-P 4-20, 5-30
-r 5-28
--receiveBufferSize 4-22
--recvBufferSize 5-30
--rprFomRevision 4-10, 5-29
--rprFomVersion 4-10, 5-29
--runCommand 5-28
-S 4-26, 5-30
-s 4-20, 5-30
--sendBufferSize 4-22, 5-30
--showConsole 5-28
--siteId 4-20, 5-30
table of 5-28
--timeConstrainingRecording 5-29
--timeRegulatingPlayback 5-29
--useEmptyFomMapper 4-11, 5-29
-v 5-28
--version 5-28
-x 4-8, 5-29
CommonEntityIds command 14-18
commonEntityTypes parameter 10-4
CommonObjectNames command 14-14
complex datatype 11-4
component files 2-4
concatenating, files 12-5, 14-21
configuration
file, RTI 2-9

configuration (continued)
window, saving and restoring 6-3
configuration file 5-23
adding string list 5-27
datatypeDefault.mtl 11-4
entity-type-dict.csv 11-12
GUI 6-6
SISO-REF-010.xml 11-12
specifying 5-28
trafficAnalyzerConfig.mtl 11-12
configuration parameter 5-2
useAsyncIO 4-21
configuring
advisory messages 14-17
application number 4-21
asynchronous I/O 4-21
data source for Microsoft Access 13-11
data source for MySQL 13-14, 13-16
DDM 5-18
DIS 4-22
heartbeat 14-19
timeout 14-19
FED file 5-29
filtering 14-14
GStreamer pipeline 5-22
high speed playback 8-7
HLA connection 14-15
indexing 5-2
list of common entity types 10-4
Logger 5-2
for HLA 4-6
Logger Remote Control 15-2
network read time 5-10
passive subscription 14-18
performance 5-10
playback, fast 5-4
Record As 14-10
recording 14-11
response to Logger Control messages 14-13
site ID 4-20
system for MAK RTI 2-8
time management 14-18
video playback 5-21
video recording 5-20
Connect command 14-12
connecting 14-12
to exercise 4-4
to network 4-3
connection
changing settings 4-4
configuring DIS 14-19
configuring HLA 14-15
editing 4-4
Logger Remote Control 15-6
properties, saving 4-4
settings 4-3
status 4-3
Connection dialog box 4-3
console, displaying 6-2
console window 14-3
displaying 5-28
Construct File Name, check box 15-3
control
Logger 6-3
remote 1-4, 15-2
controlling multiple Loggers 1-4, 15-2
ConvertFile command 14-20
converting, annotation type 9-6
conveyed region 5-14
coordinate system
traffic analyzer, changing 11-12
creating
annotation 9-2
database, MySQL 13-15
subscription control list 4-15
CSV file 11-12
cursor
setting to time 14-8
timeline 8-11
custom PDU, filtering 10-6
customPdus parameter 10-6
CutSection command 14-20
cutting
file segment 12-3
packets 14-20

D

-D command-line option 5-30
data
analyzing 11-3
delimiter, changing 11-10
exported, understanding 13-3
exporting 13-2
exporting to database 13-5
exporting to spreadsheet 13-19
importing to Excel 13-22
Data Distribution Management 5-11

data representation 11-3
data source
  Access 13-11
  configuring for ODBC 13-8
  MySQL 13-14, 13-16
data type, HLA 13-10
Data View 8-12
database
  configuring ODBC data source 13-8
  exporting data to 13-5
  exporting HLA data to 13-10
  importing to Excel 13-22
  MySQL, creating 13-15
  schema 13-5
database format, exported data 13-3
datatype 11-3, 11-6, 11-8
  complex 11-4
  mapping, changing 11-11
  supported 11-4
datatypeDefault.mtl, configuration file 11-4
Dataype Mapper, dialog box 11-11
DDM 5-11
  configuring 5-18
  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 14-21
dead-reckoning 5-9, 8-2
DeleteFile command 14-7
deleting
  annotation 9-6
  file segment 12-3
  packets 14-20
delimiter
  changing 11-4
  changing data 11-10
Delimiter Editor 11-4
Delimiter Mapper, dialog box 11-10
description
  file
    when recording 6-13
    when saving 6-14
descriptor, region 5-12
destination address 4-21, 4-22, 14-19
device address 14-19
dialog box
  Connection 4-3
  Dataype Mapper 11-11
  Delimiter Mapper 11-10
  License Setup 2-6
  Logger File Information 6-12
  New Logger File 6-13
  Record As 15-3
  Save File As 6-14
  Set Start and End Time 8-5
  Set Zoom 8-10
directory
  ./bin 2-4
  ./doc 2-4
  ./logger_tapes 2-4
DIS 1-7
  application number 4-21
  asynchronous I/O 4-21
  configuring 4-22
  configuring connection 14-19
  exercise ID 3-3, 4-20
  export to database 13-9
  filtering
    entity types 14-12
    PDUs 14-18
  heartbeat 4-24, 14-19
  port
    Logger default A-2
    specifying 5-30, A-2
  protocol version 5-30
  site ID 4-20
  timeout 4-25, 14-19
DIS Configuration page 4-24, 4-25
disabling
  attribute scaling 14-14
  backup 14-8
  entity tracking 14-7
  GUI 6-2
  high speed playback 8-7
  looping playback 14-9
  Pause on Record As 7-3
  Play On Load 8-2
  time management 14-17
  video recording 5-20
Disconnect command 14-12
disconnecting 14-12
  from exercise 6-7
displaying
  console 6-2
  console window 5-28

displaying (continued)
file information 6-12
FOM tab 11-5
object statistics 8-13
version information 5-28
docking, toolbars and plug-in views 6-5
domain 5-26
associated with command 14-21
command 14-5
drain input, maximum 5-29

E

-e command-line option 4-20, 5-30
editing
annotation 9-5
connection 4-4
file 12-2
file information 6-12
lgrConfig.xml 5-23
point annotation 9-6
range annotation 9-6
undoing changes 12-2
Editor domain, commands 14-20
empty space, trimming 12-5
enableDdm parameter 5-16
EnableRequestUpdates, command 4-18, 14-14
enabling
attribute scaling 14-14
backup 14-8
entity tracking 14-7
high speed playback 8-7
looping playback 14-9
Pause on Record As 7-3
Play On Load 8-2
time management 14-17
video recording 5-20
end
jumping to range annotation 9-7
moving to 8-11
end time
setting dynamically 9-4
setting for instant range annotation 9-3
ending, recording 7-6
end-of-file, setting 8-2
entity
common types 10-4
filtering, entity tracking 5-5
filtering by type 14-12
entity (continued)
maintenance 3-7
entity ID
filtering 5-30
filtering by 10-7
entity tracking 5-5
enabling or disabling 14-7
entity type
filtering 14-12
filtering by 10-3
mapping to string 11-12
mapping type to string 11-12
removing from filtering list 10-3
entity-type-dict.csv, configuration file 11-12
entity-type-map-file parameter 11-12
enumeration 11-6, 11-8
environment variable
MAKLMGRD_LICENSE_FILE 2-7
RTI_CONFIG 2-9
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 13-19
importing Access data to 13-22
exercise
connecting to 4-4
connection 4-3
configuring DIS 14-19
disconnecting from 6-7
ID 3-3, 4-20, 14-19
playback 3-3
playing, Logger Remote Control 15-5
recording 7-2
Logger Remote Control 15-3
Exit command 14-7
exiting, Logger Remote Control 15-7
expanding, playback segment 8-5
export, log file 13-10
exported data, understanding 13-3
ExportFile command 14-20
exporting
Access to 13-10
data, naming convention 7-5

exporting (continued)
data to database 13-5
file 13-2, 14-20
ExportSection command 14-21

F

-F command-line option 4-8, 4-10, 5-29
-f command-line option 4-12, 5-29
FDD file 2-9
features 1-2
FED file 1-7, 2-9, 4-9
changing 3-8
specifying 4-8, 4-12, 5-29
federate, specifying Logger federate name 4-7
federate type, specifying 4-7, 5-29
federation execution 1-7
name 4-8, 5-29
file
annotating 9-2
appending 12-5, 14-20, 14-21
backup 7-9, 14-8
clearing, packets 14-20
clearing segment 12-3
closing 6-10, 14-8
configuration 5-23
CSV 11-12
deleting
packet 14-20
segment 12-3
description
when recording 6-13
when saving 6-14
editing 12-2
exporting 13-2, 14-20
to database 13-5
FED 2-9, 4-12
information
displaying 6-12
editing 6-12
jumping to beginning 8-11
jumping to end 8-11
lgrConfig.xml 5-2
loading 6-7
Logger, information 6-11
merging 12-4, 14-21
offsetting 12-5
opening 14-8
file (continued)
pausing 8-2
playing 8-2
playing while recording 8-7
rclgr 15-4, 15-5
recording to multiple 7-4
repairing 6-10
reverting to saved version 12-2, 14-8
RID 2-9
saving 7-9, 14-8
segment to new file 7-10
to new name 7-9
starting 7-3
trimming 14-21
empty space 12-5
upgrading format 6-9
VR-Link.fed 4-8
filename*, logger.log* 5-19
filename command-line option 5-28
filter
entity ID 10-7
HLA object name 10-8
marking text 10-11
message 1-3
Filter By Class dialog box 10-10
Filter By Entity Type dialog box 10-3
Filter By PDU Kind dialog box 10-5
FilterEntityIds command 14-18
FilterEntityTypes command 10-4, 14-12
command, FilterEntityTypes 14-12
filterEntityTypes parameter 10-4
filtering
by entity type 14-12
by PDU kind 14-18
classes (HLA) 10-10
configuring 14-14
configuring common entity types 10-4
custom PDUs 10-6
entities 5-5
entity ID 5-30
entity type 10-3, 14-12
input and output 10-2
object names 5-29
PDU kind 10-5
PDUs 14-18
VR-Forces messages 10-12
FilterObjectNames command 14-14
FLEXlm 2-5

FOM 2-8
  database export 13-10
  flexibility 3-8
  primary 4-9
  supported 1-7
FOM Mapper 1-7, 4-11, 5-29
  initialization data 5-29
  specifying 4-12
  specifying at startup 4-10
FOM module, specifying 4-9, 5-29
FOM tab 11-2
  displaying or hiding 11-5
format, upgrading 6-9
frequency, packets for indexing 5-2
Frozen bit 5-9

G

GStreamer, configuring pipeline 5-22
GUI
  configuration file 6-6
  disabling 6-2
  localizing 2-10

H

-h command-line option 5-28
handle, start and end time 8-4
heartbeat 14-19
  defined 3-6
  HLA 5-8, 14-17
  setting 4-24
Help command 14-7
hiding
  attributes list 11-3
  FOM tab 11-5
high speed, playback 8-6
histogram 6-3, 8-12
HLA 1-7
  1516
    playback 3-4
  advisory messages 4-11, 14-17
  commands 14-14
  configuring connection 14-15
  configuring for 4-6
  configuring time management 4-14
  database schema 13-10
  DDM 5-11
  Evolved, playback 3-4
HLA 1-7 (continued)
  exporting to database 13-10
  FOM flexibility 3-8
  heartbeat 5-8, 14-17
  object name, filtering by 10-8
  passive subscription 4-17, 14-18
  specifications supported 1-7
  subscription control list 4-15
  subscription list 14-18
  support for 3-8
  time management 5-19
  timeout 5-9, 14-18
HLA Configuration page 4-14, 4-15, 4-17, 4-18
HLA Evolved
  exporting to database 13-10
  federate type 4-7
  FOM 4-8
  FOM module 4-9
  MIM 4-9
HLA Traffic Analyzer 11-2
  enabling and disabling 5-29
  options 11-9
HLA Traffic Analyzer Settings page 11-9, 11-10, 11-11, 11-12
hostname, license server 2-6

I

ID
  entity, filtering by 10-7
IEEE 1516 1-7
ignoring, advisory messages 4-11
immediate
  playback 8-2
  recording 7-2, 7-3
importing, annotation file 9-5
incomplete file, repairing 6-10
indexing, configuring 5-2
Indexing Settings page 5-3
information
  file
    displaying 6-12
    editing 6-12
  Logger file 6-11
informational message 5-19
input, filtering 10-2
inserting, file 14-21

installing
MySQL 13-14
on Linux 2-3
on Windows 2-2
RTI 2-8
instant annotation 9-3
interaction
class 11-5
parameter 11-5, 11-8
viewing 11-5
passive subscription 4-17
subscribing to 4-15
Interactions tab 11-2, 11-7
interface 1-3
text 14-3
interim PDU 3-6, 3-7
IP address 5-29

J

jumping
range annotation end 9-7
range annotation start 9-7
to beginning of file 8-11
to end of file 8-11
to specific time 8-11

K

keyboard, setting annotation times from 9-4

L

-l command-line option 5-19, 5-28
latency
video stream, reducing 8-14
layout
window, saving 6-6
lgrConfig.xml 4-4
lgrConfig.xml
configuration file 5-2
configuring high speed playback 8-7
enabling console for text interface 14-3
syntax and editing 5-23
versions for DIS and HLA 3-2
lgrControl example 1-6
lgrDump 1-6
lgrModSqlExporter example 1-6
License Manager 2-5
license server, hostname 2-6
License Setup dialog box 2-6
Linux, installing on 2-3
loading
annotation file automatically 8-2
configuration file 5-28
file 6-7, 14-8
subscription list 4-17, 14-14
LoadPlugin command 14-7
LoadSubscriptionListFile command 4-17, 5-12, 14-14
localization 1-3
localizing, GUI 2-10
log file 5-19, 5-28
sql_export.txt 13-10
logFileName parameter 5-19
Logger
controlling multiple 1-4, 15-2
recording
samples 1-5
viewing with Stealth 1-5
window 6-3
Logger command 5-2
Logger Control
configuring 14-13
message, configuring response to 14-13
Logger File Information dialog box 6-12
Logger Preferences dialog box 5-3
Logger Remote Control 15-2
command-line options 15-8
configuring Logger for 15-2
connection 15-6
exiting 15-7
playing file 15-5
rclgr file extension 15-4, 15-5
recording exercise 15-3
refresh rate 15-6
starting 15-3
loggerApp example 1-6
logger-gui.conf 6-6
logger.log 5-28
loop, enabling or disabling 14-9
looping, playback 8-2

M

-m command-line option 5-19, 5-28
macro, file naming convention 7-5

MAK RTI
  configuring 2-8
  configuring for name reservation 3-4
MAKLMGRD_LICENSE_FILE, environment variable 2-7
MAK-RPR2-1-1.fed file 1-7
mapping
  datatype, changing 11-11
  enumeration to string 11-12
marking text, filtering by 10-11
matrexTspiFomMapper example 1-6
Max Read Time 5-10
max-interaction-count parameter 11-7
MergeInFile command 14-21
merging, files 12-4, 14-21
message 14-13
  advisory 4-11
  filtering 1-3
  filtering by object name 10-8
  filtering VR-Forces 10-12
  informational 5-19
  log file 5-28
  maximum to read per tick 5-10
  VR-Link 5-19
Microsoft Access, configuring data source for 13-11
Microsoft Excel 13-19, 13-22
MIM, specifying 4-9, 5-29
Min Read Time 5-10
MOM and Initialization Module, specifying 4-9
moving
  to beginning of file 8-11
  to beginning or end of playback range 8-11
  to end of file 8-11
  to specific time 8-11
multicast address 4-26, 5-30, 14-19
multicast time-to-live 4-22, 14-19
  specifying 5-30
multiple
  files, recording to 7-4
  network card 4-23, 5-30
multiple Logger, controlling 1-4, 15-2
MySQL 13-19
  configuring data source for 13-14, 13-16
  creating database 13-15
  installing 13-14

N

-N command-line option 4-7, 5-29
-n command-line option 5-19, 5-28
name
  Logger federate 4-7
  object 11-6
name reservation 3-4
  configuring MAK RTI 3-4
naming, partial files 7-5
naming convention, for database tables and rows 13-5
network
  connection 4-3
  read time, configuring 5-10
  requirements for Logger A-2
network card, multiple 4-23, 5-30
network device 14-19
network interface address 4-22
  specifying 4-23
Network Settings page 4-4, 5-10
new file 14-10
New Logger File dialog box 6-13
NewFile command 14-10
notification level, setting 5-19
Num PDUs Per Check 5-10

O

object
  attribute, viewing 11-5
  class 11-5
  displaying statistics 8-13
  name 11-6, 14-17
  passive subscription 4-17
  subscribing to 4-15
object name
  filtering 5-29
  filtering by 10-8
Objects list 11-12
Objects tab 11-2, 11-6
ODBC, configuring data source 13-8
offset, video playback 5-21
OffsetTime command 14-21
offsetting
  file 12-5
  packets 14-21
OMT file 13-10
OpenFile command 14-8

opening
annotation file 9-5
file 6-7, 14-8
incomplete file 6-10
option, HLA Traffic Analyzer 11-9
output, filtering 10-2

P

-P command-line option 4-20, 5-30
packet
clearing 12-3
count 8-12
deleting 12-3
from file 14-20
dumping to text 13-2
offsetting 12-5
time 14-21
preventing dropped 4-21
removing
empty 12-5
from file 14-20
skipping 8-6
Packet Count View, scaling 8-13
page
Network Settings 4-4
Video Settings 5-20, 5-21
parameter
allowPlaybackWithoutRegions 5-16, 14-16
arrayBracketMap 11-13
attributeMap 11-13
commonEntityTypes 10-4
customPdus 10-6
enableDdm 5-16
entity-type-map-file 11-12
filterEntityTypes 10-4
interaction 11-8
logFileName 5-19
max-interaction-count 11-7
playbackDynamicAssociation 5-16
playbackDynamicExtents 5-16
recordConveyedRegions 5-16
requestUpdates 5-16
showConsole 14-3
subscribeMode 5-11
SubscribeRegionsAreGlobal 5-11
SubscribeRegionsAreSpecified 5-11, 5-17, 14-17
subscribeUnspecifiedAttributes 5-13, 5-17, 14-17
useAsyncIO 4-21
viewing interaction 11-5
partial file, naming convention 7-5
passive subscription 4-17
configuring 14-18
path, specifying for RTI 2-8
Pause command 14-8
Pause on Record As 7-3
PauseRecording command 14-10
pausing
playback 5-9, 8-2, 14-8
recording 7-6, 14-10
PDU
filtering 14-18
filtering by 10-5, 14-18
filtering custom 10-6
maximum to read per tick 5-10
supported for database export 13-9
performance, configuring 5-10
pipeline
GStreamer, configuring 5-22
Play button 8-2
Play command 14-8
Play On Load 6-8, 8-2
playback 3-3, 6-7
automatic 8-2
bounding to annotation 9-9
DDM 5-15
described 3-3
exercise ID 3-3
fast 5-4
faster, configuring 14-7
high speed 8-6
HLA 1516 3-4
HLA Evolved 3-4
loading file 6-7
looping 8-2, 14-9
mode 3-3
moving to beginning 8-11
moving to end 8-11
pausing 5-9, 8-2, 14-8
range, moving to beginning or end 8-11
segment 8-3, 8-4, 8-5
expanding 8-5
setting speed 8-6, 14-9
starting 14-8
stopping 8-3, 14-9
time regulating 5-29
time-managed 5-19
unpausing 14-9
while recording 8-7
playbackDynamicAssociation parameter 5-16

playbackDynamicExtents parameter 5-16
playbackMode 5-16, 14-16
PlaybackRegionNotEnabled 5-16
PlaybackRegionsUseConveyedExtents 5-16
PlaybackRegionsUseData 5-16, 14-16
Player domain, commands 14-8
playing
  file 8-2
  part of a file 8-3, 8-4, 8-5
PlayPause command 14-8
plug-in 6-3
  docking and undocking views 6-5
  example 1-6
  object statistics 8-13
point annotation
  adding instantly 9-3
  converting to range 9-6
  editing 9-6
Point Annotation dialog box 9-5
port 14-19
  specifying UDP 4-20
  video playback 5-21
preferences, setting 5-2
PrefixFileToStart command 14-21
prefixing, file 14-21
property
  annotation 9-5
  connection 4-3
protocol
  supported 1-7
  version, DIS 5-30

Q

Qt Linguist 2-10

R

-r command-line option 5-28
range annotation
  adding instantly 9-3
  bounding playback to 9-9
  converting to point 9-6
  editing 9-6
  jumping to end 9-7
  jumping to start 9-7
  setting end time 9-4
Range Annotation dialog box 9-5
rclgr file extension 15-4, 15-5
receive buffer 4-22, 14-19
  size, specifying 5-30
record, mode 3-3
Record As, configuring 14-10
Record As dialog box 15-3
Record command 14-10
recordConveyedRegions parameter 5-16
Recorder domain, commands 14-10
recording 3-3
  and playing 8-7
  automatically 7-3
  canceling 14-10
  closing 14-10
  configuring 14-11
  described 3-3
  exercise 7-2
    ID 3-3
    Logger Remote Control 15-3
  immediately 7-2
  multiple files 7-4
  pausing 7-6, 14-10
  repairing 6-10
  resuming 7-6
  saving 7-9
  starting 7-3
  stopping 7-6
  time constrained 5-29
  time-managed 5-19
RecordToMultipleFiles command 14-10
refresh rate, Logger Remote Control 15-6
region
  conveyed 5-14
  subscription 5-11
region descriptor 5-12
region set 5-12
remote, control 1-4, 15-2
remote control ID 15-3
  specifying 5-28, 14-12
Remote Control Logger 1-4
removing
  dead space 14-21
  empty packets 12-5
  packets from file 14-20
repairing, incomplete file 6-10
requesting, attribute value updates 4-18
requestUpdates parameter 5-16
resetting, annotation time point 9-4
Restart command 14-8
restoring, window configuration 6-3
resuming, recording 7-6

RevertFile command 14-8
reverting
  file 14-8
  to saved file 12-2
RID file 2-9
rid.mtl file 2-9
row, naming conventions 13-5
RPR FOM 3-8, 4-10
  revision, specifying at startup 4-10
RTI 2-8
  definition of 2-8
  specifications supported 1-7
RTI_CONFIG environment variable 2-9

S

-S command-line option 4-26, 5-30
-s command-line option 4-20, 5-30
sample recording 1-5
sampleFilter example 1-6
Save 7-9
Save As 7-9, 7-10
Save File As dialog box 6-14
SaveFile command 14-8
SaveFileAs command 14-8
SaveSubscriptionListFile command 4-16, 14-15
saving
  annotation file 9-4
  connection properties 4-4
  file 14-8
    to new name 7-9
  files 7-9
  segment as new file 7-10
  subscription list 4-16, 14-15
  window layout 6-6
scaling
  Packet Count View 8-13
  time 3-7
schema, database 13-5
segment
  playing 8-3, 8-4, 8-5
  saving as new file 7-10
  setting playback to annotation 9-9
selecting, annotation 9-5
send buffer 4-22, 14-19
  size, specifying 5-30
sending, attribute value update 4-18
set, region 5-12
Set Start and End Time dialog box 8-5
Set Time 3-7, 8-11
Set Zoom dialog box 8-10
SetBackChannelConnection command 14-12
SetBackupOnSave command 7-9, 14-8
SetCheckpointParameters command 5-4, 14-10
SetCommonEntityTypes command
  command, SetCommonEntityTypes 14-12
SetConnectionParameters command 4-6, 4-19, 14-12, 14-15, 14-19
SetCurrentTime command 14-8
SetDdmParameters, command 14-16
SetDdmParameters command 5-11, 5-13, 5-15
SetDefaultTapeDirectory command 14-7
SetDisableTimeManagementTest command 14-17
SetDrainInput command 14-17, 14-19
SetDynamicNamePrefix command 3-3, 14-17
SetEntityTracking command 5-4, 5-5, 14-7
SetFastJumpsWithoutDiff command 5-4, 5-6, 14-7
SetFollowingExportInterval, command 14-9
SetFollowingInterval, command 14-9
SetGuiConfigFilename command 6-6, 14-7
SetHeartbeatIntervals, command 5-4, 5-8, 14-17
SetHeartbeatRate command 4-24, 14-19
SetIgnoreAdvisories command 4-11, 14-17
SetListenWhenIdle, command 14-12
SetLoopAtEnd command 14-9
SetNotifyLevel command 5-19, 14-7
SetPacketViewInterval, command 14-9
SetPassiveSubscription command 4-17, 14-18
SetPauseOnRecordAs command 14-10
SetPlaybackReadTimeLimit command 5-4, 5-7, 14-7
SetPlaybackSpeed command 14-9
SetPlayOnLoad command 14-9
SetRecordToMultipleFilesSettings command 14-10, 1-2
SetRemoteControlId command 14-12, 14-13
SetRemoteControlSettings command 14-13

SetSaveConnectionConfiguration, command 14-14
SetScaleAttributes command 14-14
SetSendOnlyUpdatedValues, command 4-19, 14-14
SetSkippingPlaybackThreshold command 14-9, 1-1
SetStartStopTimes command 14-9
SetStopCondition command 14-11
SetStopDuration command 14-11
SetStopSize command 14-11
SetSubscriptionLists command 4-16, 5-27, 14-18
SetTempDir command 14-7
SetTimeManagement command 4-14, 14-18
SetTimeoutIntervals, command 5-4, 5-9, 14-18
SetTimeoutRate command 4-25, 14-19
setting
  checkpoint policy 14-10
  cursor to time 14-8
  DIS heartbeat 4-24
  DIS timeout 4-25
  end-of-file action 8-2
  notification level 5-19
  playback segment to annotation 9-9
  playback speed 8-6, 14-9
  preferences 5-2
SetVRFFiltering command 14-14
showConsole parameter 14-3
simulation
  recording 7-2
  standard, supported 1-7
Simulation domain, commands 14-12
SISO 1516 API 1-7
SISO enumeration XML document 11-12
SISO-REF-010.xml, configuration file 11-12
site ID 4-20, 5-28, 14-12, 14-19
  specifying 5-30
size, indexing 5-2
skip command, entity maintenance 3-7
Skip Time 3-7
skipping, packets 8-6
skippingPlaybackThreshold 8-7
SkipTime command 14-9
slider
  start and end time 8-4
  zoom 8-4
slow-speed playback 3-6
specifying
  application number 5-30
  configuration file 5-28
  FED file 4-8, 4-12, 5-29
  federate type 4-7, 5-29
  federation execution name 4-8
  FOM Mapper 4-12
  FOM modules 4-9, 5-29
  MIM 4-9, 5-29
  MOM and Initialization Module 4-9
  multicast time-to-live 5-30
  network interface address 4-23
  receive buffer size 5-30
  remote control ID 5-28, 14-12
  send buffer size 5-30
  site ID 5-30
  subscription list 14-18
  subscription region 5-12
  time management 5-19
  UDP port 4-20
speed
  control 6-3
  setting for playback 8-6
  setting playback 14-9
speed factor, video playback 5-21
spreadsheet, exporting data to 13-19
SQL, exporting data to 13-5
sql_export.txt log file 13-10
start
  jumping to range annotation 9-7
  moving to 8-11
start time, setting dynamically 9-4
starting
  Logger 6-2
  Logger Remote Control 15-3
  playback 14-8
  recording 7-3
startup
  playback mode 3-3
  record mode 3-3
  running commands 5-28
  UNIX 6-2
  Windows 6-2
stateObserver example 1-6
status, field 7-2
stopping
  playback 8-3, 14-9
  recording 7-6
StopPlayback command 14-9

StopRecording command 14-11
string list 4-16
  adding to configuration file 5-27
subnet mask 4-22, 14-19
subscribeMode 5-16
subscribeMode parameter 5-11
SubscribeRegionNotEnabled 5-16, 14-16
SubscribeRegionsAreGlobal 5-16
SubscribeRegionsAreGlobal parameter 5-11
SubscribeRegionsAreSpecified 5-16
SubscribeRegionsAreSpecified parameter 5-11, 5-17, 14-17
subscribeUnspecifiedAttributes parameter 5-13, 5-17, 14-17
subscribing, passively 4-17
subscription, region 5-11
subscription control list, creating 4-15
subscription list
  DDM, setting programmatically 5-14
  loading 4-17, 14-14
  saving 4-16, 14-15
  specifying 14-18
subscription region, specifying 5-12
syntax, *lgrConfig.xml* 5-23
System domain, commands 14-7

T

table, naming conventions 13-5
tape
  annotating 9-2
  editing 12-2
  loading 6-7
TCP 5-30
temporary file, recording to 7-2
text, dumping file to 13-2
text interface 14-3
time
  command, entity maintenance 3-7
  indexing 5-2
  jumping to 8-11
  navigating during playback 8-11
  navigation 8-11
  offsetting packet 14-21
  setting 8-11
  setting cursor to 14-8
  sliders 8-4
time constrained, recording 5-29
time jump, updating entities after 3-7
time management
  configuring 14-18
    in GUI 4-14
  enabling or disabling 14-17
  specifying use of 5-19
time regulating, playback 5-29
time scale
  described 3-4, 3-5, 3-6
  during recording 7-2
  effects 3-6
timebar
  color, changing 15-7
time-constrained, setting 4-14
timeline
  cursor 8-11
    moving 8-11
  zooming 8-9
    to annotation 9-8
timeout 14-19
  defined 3-6
  HLA 5-9, 14-18
  setting 4-25
timepoint, setting 8-11
time-regulating, setting 4-14
timestamp, offsetting 12-5
Toggle Attributes Pane bar 11-3
toolbar 6-3
  docking and undocking 6-5
Traffic Analyzer 11-2
  array elements, brackets 11-13
trafficAnalyzerConfig.mtl, configuration file 11-12
translating, GUI text 2-10
translation 1-3
trimming, file 12-5, 14-21
TrimStart command 14-21
type, object 11-6

U

UDP port 5-30
  specifying 4-20, 5-30
undo, edits 12-2
undocking, toolbars and plug-in views 6-5
UNIX
  starting, Logger 6-2
Unpause command 14-9
UnpauseRecording command 14-11
unpausing 14-9
Update Annotation 9-4

updating, annotation 9-4
Upgrade File Format 6-9
upgrading, file format 6-9
useAsyncIO parameter 4-21

V

-v command-line option 5-28
variable
  adding to *lgrConfig.xml* 5-26
  syntax in *lgrConfig.xml* 5-25
variant 11-6, 11-8
VCR controls 6-3
velocity, scaling 3-7
version, displaying 5-28
video
  configuring GStreamer pipeline 5-22
  offset for playback 5-21
  playback, configuring 5-21
  port, playback 5-21
  speed factor 5-21
  viewing 8-3
video recording, enabling and disabling 5-20
Video Settings page 5-20, 5-21
video stream, reducing latency 8-14
View Annotations window 9-2
view control PDUs 1-6
View menu 6-4
viewing
  connection properties 4-3
  interaction parameter 11-5
  object attribute 11-5
  object statistics 8-13
  video 8-3
VLC
  reducing latency 8-14
  viewing streaming video in 8-3
VR-Forces message 10-12
VR-Link, messages 5-19
VR-Link.fed file 4-8
VR-TheWorld_FirstExperienceDIS 1-5

W

window
  configuration 6-3
  layout, saving 6-6
  Logger 6-3
Windows
  installing on 2-2
  starting Logger 6-2

X-Y-Z

-x command-line option 4-8, 5-29
XML, RTI configuration file 2-9
XML file 13-10
zoom area 8-4
zooming
  Packet Count View 8-13
  timeline 8-9
  timeline to annotation 9-8

MAK Data Logger Quick Reference Card

Table 1: Logger commands

Command	Description
System domain commands	
`DeleteFile` *filename*`(string)`	Deletes the specified file.
`Exit`	Shuts down the Logger.
`Help`	Displays a list of available commands.
`LoadPlugin` *pluginName*`(string)`	Loads the specified plug-in.
`SetDefaultTapeDirectory` *defaultTapeDir*`(string)`	Sets the directory where the Logger creates and loads files.
`SetEntityTracking` *entityTracking*`(bool)`	Allows entity tracking to be processed.
`SetFastJumpsWithoutDiff` *fastJumpsWithoutDiff*`(bool)`	Uses checkpoints to establish the relative state of objects, but the packets between the checkpoint and jump time are not processed
`SetGuiConfigFilename` *filename* `(string)`	Specifies the name of the GUI layout configuration file. Default: *logger-gui.conf*.
`SetNotifyLevel` *vlLevel*`(int)` *lgrLevel*`(int)`	Sets the notification level of VR-Link and the Logger.
`SetPlaybackReadTimeLimit` *playbackReadTimeLimit*`(double)`	Limits how long the Logger can spend reading packets during playback.
`SetTempDir` *directory*`(string)`	Sets the directory in which the Logger stores temporary Logger files.
Player domain commands	
`ClosePlaybackFile`	Closes the current playback file.
`OpenFile` *filename*`(string)` *constructFilename* `(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` *reopenAfterSave*`(bool)`	Saves a previously saved, but currently unsaved file to its current file name.
`SaveFileAs` *filename*`(string)` *description*`(string)` *writeDescription*`(bool)` *reopenAfterSave*`(bool)`	Saves the current file to new file.
`SetBackupOnSave` *backup*`(bool)`	Enables or disables automatic backup of previously saved files when saving edited files.
`SetCurrentTime` *time*`(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` *value*`(bool)`	Enables or disables looping to start at the end of playback.
`SetPacketViewInterval` *interval* `(double)`	Sets the interval between tick marks in the Packet Count View and Packet Size View.
`SetPlaybackSpeed` *speed*`(double)` *skippingPlayback*`(bool)`	Sets the playback speed of the Logger.
`SetPlayOnLoad` *value*`(bool)`	Enables starting playback immediately after opening a file.
`SetSkippingPlaybackThreshold` *skippingPlaybackThreshold*`(double)`	Set the playback speed above which skipping playback mode is used automatically.

Copyright © 2021 MAK Technologies. All rights reserved. Document ID: LOG-5.7-1QRC-200722

Table 1: Logger commands

Command	Description
SetStartStopTimes *startTime*(time) *stopTime*(time)	Sets the start and stop time of the Logger.
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 playback 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 *filename*(string) *description*(string) *constructFilePathAndName*(bool)*overwrite*(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 *fileName*(string)	Starts recording to new files.
SetCheckpointParameters *policy*(int) {*size*(int) | *count*(int) | *time*(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 *record*(bool)	Determines how the Logger behaves when given the Record As command.
SetRecordToMultipleFilesSettings *duration*(double) *namingConvention*(string)	Set the multiple file recording parameters.
SetStopCondition *condition*(int)	Specifies the stop condition for recording as follows: ◆ 0 — no condition. ◆ 1 — duration (enables SetStopDuration). ◆ 2 — file size (enables SetStopSize).
SetStopDuration *duration*(time)	Stops recording at the specified time if SetStopCondition is set to 1.
SetStopSize *filesize*(int64)	Stops recording when the file reaches *filesize* if SetStopCondition is 2.
StopRecording *closeAndOpenForPlayback*(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 *entityType1 ... entityTypen*(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 *useBackChannel*(bool) *address*(string) *port*(int) *interface*(string)	Set the parameters for the back channel connection configuration.
SetCommonEntityTypes *entityType1 ... entityTypen*(string)	Specifies a list of entity types to list in the Filter by Entity Type dialog box.
SetListenWhenIdle *state* (bool)	Enable or disable listening to the network when not recording or playing.
SetRemoteControlSettings *process*(bool) *process*(bool) *recordskippingPlayback*(bool) *allowDelete*(bool) *allowAll*(bool) *sendResponses*(bool) *crippleGui*(bool) *useLegacyPdu*(bool) *connectSendInterval*(double) *heartbeatSendInterval*(double)	Specifies how the Logger responds to Logger Control messages.
SetSaveConnectionConfiguration *saveConnectionConfiguration* (bool)	Enables or disables saving of connection information to *lgrConfig.xml*.
SetScaleAttributes *scale*(bool)	Enables or disables attribute scaling due to changes in playback speed.

MAK Data Logger Quick Reference Card

Table 1: Logger commands

Command	Description
`SetSendOnlyUpdatedValues` *state* (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` *filter*(bool) *loId*(int) *hiId*(int)	Configures VR-Forces message filtering.
HLA-specific commands	
`CommonObjectNames` *objectName1 ... objectNameN*(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` *state* (bool)	Enable or disable whether request updates is invoked when an object is discovered. Default: true.
`FilterObjectNames` *objectName1 ... objectNameN*(string)	Specifies a list ofobject names to list in the Filter by Object Names dialog box and enables filtering for these entity object names.
`LoadSubscriptionListFile` *filename*(string)	Loads a subscription list file.
`SaveSubscriptionListFile` *filename*(string)	Saves the subscription settings to an XML file with the specified name.
`SetConnectionParametersHLA` *federateType*(string) *federationName*(string) *fedFile*(string) *ignoreAdvisories*(bool) *fomVersion*(double) *fomRevision*(int) *fomMapper*(string) *fomMapperData*(base64) *useEmptyFomMapper*(bool) *useMatrexFomMapper*(bool) *matrexFomMapperData*(string) `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)	Specifies HLA (1.3 and 1516) or HLA Evolved connection parameters.
`SetDdmParameters` *enableDdm playbackMode subscribeMode playbackDynamicAssociation playbackDynamicExtents allowPlaybackWithoutRegions recordConveyedRegions requestUpdates subscribeUnspecifiedAttributes*	Specifies DDM parameters.
`SetDisableTimeManagementTest` *disableTesting*(bool)	Disables time management testing if it causes the Logger to misbehave.
`SetDrainInput` *maxTime*(double), *minTime*(double)	Specifies the minimum and maximum amount of time, in seconds, to spend reading messages from the network each tick.
`SetDynamicNamePrefix` *prefix* (string)	Specifies the prefix that the Logger attaches to the beginning of object names that begin with HLA_ when it is recording exercises.
`SetHeartbeatIntervals "`*className*(string):*interval*(float):*inheritable*(bool) ..."	Specifies a heartbeat interval for playback. You can specify multiple classes.
`SetIgnoreAdvisories` *mode*(bool)	Specifies the Logger's response to HLA advisory messages. Default: false.
`SetPassiveSubscription` *mode*(bool)	Configures passive subscription for HLA. Default: false.
`SetSubscriptionLists` *objectClasses*(stringList) *interactionClasses*(stringList)	Sets the class subscription list.
`SetTimeManagement` *timeConstrainedRecording*(bool) *timeConstrainedPlayback*(bool) *timeRegulatingRecording*(bool) *timeRegulatingPlayback*(bool) *lookahead*(double)	Configures time management options.
`SetTimeoutIntervals "`*className*(string):*interval*(float):*inheritable*(bool) ..."	Specifies an object timeout for recording. You can specify multiple classes.

Copyright © 2021 MAK Technologies. All rights reserved. Document ID: LOG-5.7-1QRC-200722

Table 1: Logger commands

Command	Description
DIS-specific commands	
AddCustomPDU *pduName*(string) *pduKind*(string) *entityIdOffset*(string) *entityTypeOffset*(string)	Specifies a PDU to add to the Filter by PDU Kind dialog box.
CommonEntityIds *id1 ... idN*(string)	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box. Does not enable filtering of these entity IDs.
FilterEntityIds *id1 ... idN*(string)	Specifies a list of entity IDs to list in the Filter by Entity IDs dialog box and enables filtering for these entity IDs.
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)	Specifies parameters for the DIS exercise connection.
SetDrainInput *maxTime*(double), *numPdusToRead*(int)	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.
SetHeartbeatRate *rate*(double)	Specifies the DIS heartbeat rate.
SetTimeoutRate *rate*(double)	Specifies the DIS timeout rate.
Editor domain commands	
AppendFileToEnd *filename*(string)	Appends the file *filename* to the end of the open file.
ClearSection *start*(time) *end*(time)	Removes all packets between the *start* and *end* times in a file.
ConvertFile	Converts the current file to the latest file format.
CutSection *start*(time) *end*(time)	Removes all packets between the *start* and *end* times in a file.
ExportFile *exportFormat*(string) *destination*(string) *checkpoint*(bool) *openAfterExport*(bool)	Exports a file using a given *exportFormat* to a *destination*.
ExportSection *exportFormat*(string) *destination*(string) *start*(time) *end*(time) *offset*(double) *checkpoint*(bool) *openAfterExport*(bool)	Exports a section of a file using a given *exportFormat* to a *destination*.
MergeInFile *filename*(string) *offset*(double)	Merges the Logger file *filename* into the open file.
OffsetTime *offset*(double)	Offsets the time of the packets in the open file by the *offset* value.
PrefixFileToStart *filename*(string)	Inserts the file named *filename* at the start of the open file.
TrimStart	Removes any dead time from the start of the file and offsets all packets so that the first packet time stamp is 0.0.

MAK Data Logger Quick Reference Card

Table 2: 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	Enter
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	Enter	Trim Start	Ctrl + Shift + T
Object Statistics	Shift + O	Update Annotation	]
Offset Time	Ctrl + Shift + O	Upgrade File Format	Ctrl + Shift + U

Copyright © 2021 MAK Technologies. All rights reserved. Document ID: LOG-5.7-1QRC-200722

Command-line Options

Option	Description
`--allowAllRemoteControl` *mode*	Allow (1) or do not allow (0) remote control PDUs.
`-c` *mode*	If *mode* = 1, show the Logger console.
`--configFile` *filename*	Specifies the configuration file to load.
`--defaultTapeDir` *directory*	Specifies the default directory for Logger files.
`--filterEntityTypes`	Adds entity types to Filter by Entity Type dialog box and selects them.
`--ignore_rest`	Ignore all arguments after this one.
`-h`	Displays a brief summary of Logger command-line arguments.
`-l` *logfile*	Specifies a file to send notification messages to. Default: *logger.log*.
`--loggerTickRate` *rate*	The interval for the Logger tick rate, in milliseconds.
`-m` *level*	Specifies the notification level for Logger messages. Range 0-4.
`-n` *level*	Specifies the notification level for VR-Link messages. Range 0-4.
`-r` *command*	Specifies a Logger command to run. Can be used multiple times.
`--remoteControlId` *ID*	Specifies the Logger's remote control ID. Default: site ID.
`-v`	Displays version information and exits.
HLA Only	
`--disableTMTesting` *mode*	Disable time management for testing.
`--enableTrafficAnalyzer` *mode*	Enable (1) or disable (0) the HLA Traffic Analyzer.
`-f` *libname*	Specifies the FOM Mapper to use if you are not using the default FOM.
`--federateType` *type*	Specifies the federate type for the Logger federate.
`--filterObjectNames` *names*	Filter the listed object names. Accepted multiple times.
`--fomMapperData` *data*	Specifies FOM Mapper initialization data, if required.
`--fomModules` *module*	Specifies a FOM module for the exercise. (accepted multiple times, HLA Evolved only)
`-F` *fedfile*	Specifies the FED file for the exercise.
`--ignoreAdvisories` *mode*	If 1, ignore HLA advisory messages. If 0, do not ignore.
`--maxDrainInputTime time`	Specifies the maximum time to drain input.
`--MIM` *MIM_file*	Specifies the MIM for the exercise. (HLA Evolved only)
`-N` *federate_name*	Specifies the federate name for the Logger federate. (HLA Evolved only)
`--rprFomVersion` *version*	Specifies the version of the RPR FOM to use.
`--rprFomRevision` *revision*	Specifies the FOM revision to use.
`--ignoreAdvisories {0|1}`	Specifies whether (1) or not (0) the Logger ignores advisory messages.
`--timeConstrainingRecording` *mode*	Use time constrained for recording.
`--timeRegulatingPlayback` *mode*	Use time regulating for playback
`--useEmptyFomMapper {0|1}`	Specifies whether (1) or not (0) the Logger uses an empty FOM Mapper.
`-x` *ExName*	Specifies the federation execution name. Default: VR-Link.
DIS Only	
`-A` *address*	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.
`-a` *appnumber*	Specifies the application number.

MAK Data Logger Quick Reference Card

Command-line Options

Option	Description
(-D | --deviceAddString) *address*	Specifies the network interface address.
--disVersion *num*	Sets the DIS protocol version for PDUs sent onto the network. *num* can be 4 (DIS 2.0.4), 5 (IEEE 1278.1), or 6 (DIS 2.1.4, IEEE 1278.1a).
(-e | --exerciseId) *ID*	Sets the exercise ID.
--filterEntityIds *IDs*	Filter the specified entity IDs. Acceptable multiple times.
--mcastTtl *ttl*	Specifies the multicast time-to-live.
-P *port*	Specifies which DIS port (UDP port if -A not specified, otherwise, TCP) the Logger uses to send or receive data. Default: 3000.
--recvBufferSize *size*	Specifies the receive buffer size.
--sendBufferSize *size*	Specifies the send buffer size.
-s *ID*	Specifies the site ID.

Copyright © 2021 MAK Technologies. All rights reserved. Document ID: LOG-5.7-1QRC-200722

LOG-5.9-1-210812