MÄK Logger

The Data Recorder / Player
for the Virtual World

MÄK TECHNOLOGIES

MÄK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138 USA

Second edition: October 1998

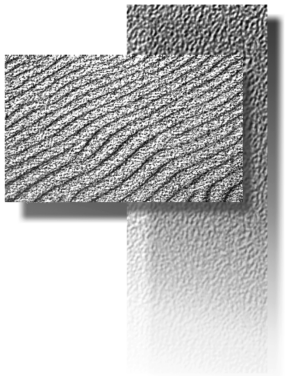
MÄK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138 USA

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

info@mak.com

www.mak.com

Revision LOG-3.3-1-981007



Contents

Chapter 3 Logger Concepts

Chapter 4 Using The Logger Window

Chapter 5 Using The TLogger

Preface

The MÄK Logger is a data recorder that records HLA and DIS activity and plays it back for review. The Logger gives you a choice of a graphical VCR-style interface or a shell-style command line interface.

This manual is for persons who will install the Logger and for persons who will record or play back simulations. It assumes that you are familiar with basic UNIX or DOS operations and that you know how to work in your computer environment, either OSF/Motif or Windows.

The Logger is a dynamic product, constantly evolving to meet user needs. Consequently, you may find improvements or new features that have been added to the Logger since this manual went to press.

How the Manual is Organized

The manual is organized as follows:

Chapter 1, *Introduction*, describes the Logger and provides an introduction to simulation protocols.

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

Chapter 3, *Logger Concepts*, provides an overview of what the Logger does, and describes the Logger's features. This chapter also explains advanced processing features which may not be obvious during Logger operation.

Chapter 4, *Using The Logger Window*, shows how to record and play back HLA/DIS network activity using the window-based versions of the Logger.

Chapter 5, *Using The TLogger*, shows how to use the TLogger, the text-based command line version of the Logger.

Chapter 6, *Controlling the Logger Remotely*, explains how to control the Logger programmatically.

Chapter 7, *Logger Tools*, describes the utility programs that combine, modify and inspect existing Logger recordings.

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

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

MÄK Products for the Virtual World

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

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

- The VR-Link Network Toolkit. The Toolkit is an object oriented library of C++ functions and definitions that implement the HLA RPR FOM and the DIS protocol. This library minimizes the time and effort required to build and maintain new HLA/DIS-compliant applications, and to integrate such compliance into existing applications.

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

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

- The MÄK Stealth. The Stealth displays a realistic, three-dimensional view of your virtual world. You can view this world from the inside of a simulated moving vehicle, or place the eyepoint at another moving or stationary location. The Stealth lets you switch rapidly among several predefined viewpoints while the simulation is underway.
- The MÄK Plan View Display. The Plan View Display (PVD) provides a two-dimensional, “big picture” of a virtual environment. It shows simulated entities, such as vehicles, soldiers, and tanks moving over a 2D-map display. The MÄK PVD is compatible with both Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) simulation standards.

- Dial-a-Sim. Dial-a-Sim is a reconfigurable DIS simulator program that lets you create and simulate vehicles quickly, with or without programming. Dial-a-Sim's graphical interface lets you assemble vehicles by choosing from a collection of predefined components. You can adjust the physical properties and positioning of these components to meet your exact needs.

  Once your vehicle is complete, you can drive it in a networked simulation. Dial-a-Sim provides a set of on-screen controls, and supports PC-compatible steering wheel assemblies and other hardware controls.

- The MÄK RTI. The MÄK RTI is the first commercial run-time infrastructure for HLA simulations.

Getting Technical Support

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

For sales and upgrade information by e-mail:	vrlink-info-sales@mak.com
For technical support by e-mail:	vrlink-tech-spt@mak.com
For general MÄK information via the Web:	http://www.mak.com

Send postal correspondence to:

MÄK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138

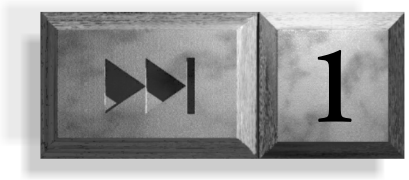
Call or fax us at:	Fax:	617-876-9208
	Voice Phone:	617-876-8085

Document Conventions

This document uses the following typographic conventions:

`Monospaced Type`	Indicates commands or values you enter.
`Monospaced Italic`	Indicates variables that you replace with appropriate values.
`[ ]`	Indicates optional arguments.
`|`	Separates options in a command where only one option may be chosen at a time.
/	Indicates a directory. Since MÄK products run on both UNIX and PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
Bold type	Indicates a new term.
Italic	Indicates a file name or path, or a class name.
sansSerif	Indicates a parameter or argument.
>	Indicates a one-step procedure.
Menu→Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose File→Save.

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



Introduction

This chapter covers the following topics:

1.1. Logger Features

The MÄK Logger is a data recorder that records HLA and DIS activity and plays it back for review. The Logger gives you a choice of a graphical VCR-style interface, a shell-style command line interface, and programmatic control.

All versions of the Logger let you scale and reverse time during playback. You can quickly scan a Logger recording to find an area of interest, and then replay that section forward or backward at any speed.

The Logger is available for a range of UNIX platforms as well as for Windows NT and Windows 95.

The Logger package includes the following:

- The Logger, an HLA/DIS data-recorder/player with a VCR-style graphical user interface.
- The TLogger (or Text Logger), which performs the same functions as the Logger, but has an interactive, character-based shell-style interface. The TLogger adds script support to the Logger's feature set.
- A set of Logger Tools that let you combine, modify and inspect recorded Logger files. You can merge the activity from several exercises[1] into a single Logger file, extract a portion of an exercise to a new file, and perform other file operations.
- A set of sample Logger recordings (also called Logger tapes).

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

1.2. A Brief Introduction to DIS and HLA

Distributed Interactive Simulation (DIS) and High-Level Architecture (HLA) are means of representing players in a shared virtual world through the exchange of data between applications. These applications allow user interaction and usually run on separate, networked computers.

1.2.1. The DIS Protocol

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.

1.2.2. The HLA Protocol

HLA is a relatively new protocol that has been adopted as the standard technical architecture for all Department of Defense simulations. It allows more flexibility than DIS. The Run-Time Infrastructure (RTI) determines the wire formats and mechanisms for data delivery. Each individual simulation domain can develop its own Federation Object Model (FOM), which defines the types of information to be shared among simulators.

The Realtime-Platform Reference (RPR) FOM is based on the DIS protocol, and is therefore especially important to developers migrating applications from DIS to HLA. The Logger supports the FOM defined within the Federation Execution Data (FED) file called *vrlink.fed* stored in the Logger home directory. See the Release Notes for your version of the Logger for complete information about the FOMs supported by the Logger.

1.2.3. Obtaining Further Information about HLA and DIS

For information about HLA rules and specifications, visit the HLA Web site at:

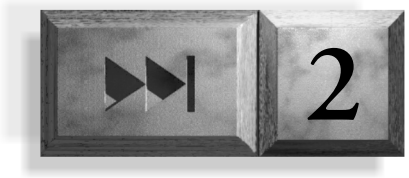
http://hla.dmso.mil/

You can find general DIS information in the document library at the Simulation Interoperability Standards Organization (SISO) home page:

http://www.sisostds.org/doclib/

For documentation of the IEEE 1278.1-1995 and 1278.2-1995 versions of DIS, obtain the publication, *1278.1-1995 IEEE Standard for Distributed Interactive Simulation, Application Protocols*. For more information, visit the IEEE World Wide Web site at:

http://stdsbbs.ieee.org/products/catalog/catalog.html



Installing the MÄK Logger

This chapter covers the following topics:

2.1. System Requirements

This section lists hardware and software requirements for running the Logger.

2.1.1. Disk Space Requirements

The Logger requires about 2.5 Mb of hard disk space, plus an additional 33Mb for the sample Logger recordings. If you want to save space on your hard disk, you can choose not to install the recordings and play them directly from the CD.

2.1.2. Supported Platforms

The Logger is available for the platforms listed below.

Note: The HLA version works only on platforms for which an RTI is available.

- Silicon Graphics workstations with Irix6.x.
- Sun Sparcstation with SunOS 4.x or Solaris 2.4 (or later).
- DEC Alpha with OSF/1 v4.0.
- HP 9000 Series 700 with HPUX operating system version 9.
- Windows NT or Windows 95 with the TCP/IP protocol installed and a 486, Pentium, or Pentium Pro-class CPU.

2.1.3. HLA Support

This section describes compatibility and support issues for running the Logger in HLA. HLA RTIs and FOMs are undergoing rapid change as the HLA matures. Please refer to the *RELNOTES* file and any supplementary documentation for the latest release of the Logger for current information about FOM support.

FOM Flexibility

Logger 3.2 or later reads the *.fed* file, and subscribes to all non-MOM classes in the file. Therefore, you can modify or extend the FOM and the Logger will record and play back your new data.

Logger 3.2 or later allows you to 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, your file can be played back successfully.

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 will not appear in outgoing updates and interactions.

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 *RELNOTES* file and any supplementary documentation for the latest release of the Logger for current information about backward compatibility.

2.2. Installing the Logger

A complete Logger installation involves installing the Logger itself, the FLEXlm license manager, and the HLA Runtime Infrastructure (RTI). These tasks are described separately over the following pages.

2.2.1. Installing the Logger on UNIX

1. Mount the CD-ROM and run:

   ```
   ./install
   ```

2. Follow the on-screen instructions.

You can install the Logger into any directory for which you have write permissions.

2.2.2. Installing the Logger in Windows

1. Insert the MÄK PC Software CD-ROM into your CD-ROM drive.
2. From the Windows Explorer, run the *Setup.exe* program located in the root directory of the CD-ROM.
3. On the opening screen, select the icon for the Logger.
4. Follow the on-screen instructions. You will be prompted to confirm the directory where the Logger is to be installed. You can replace the suggested default with a different destination.

   If you do not install the sample Logger tapes, you can copy them to the disk later or play them directly from the CD.

2.2.3. The Logger Directory Structure

The Logger installation process creates a directory structure like the one shown in Figure 2-1.

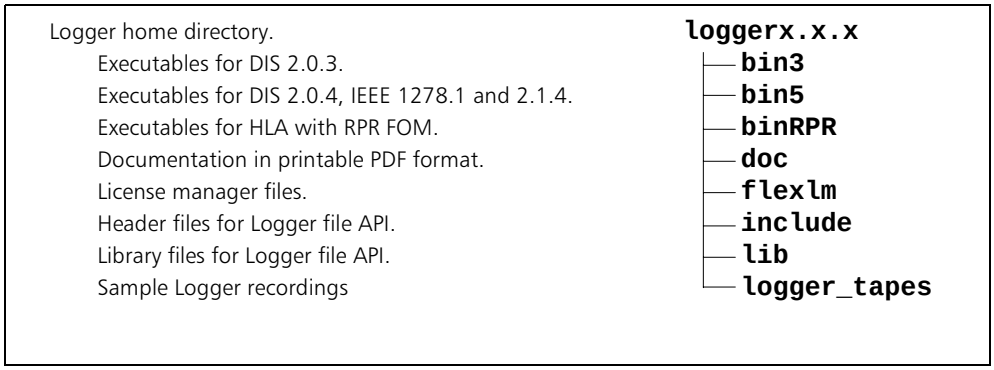


Figure 2-1. Destination of Logger files

The Binary Directories

This package contains three sets of executables in order to support a range of different protocols. The *./binRPR* directory contains executables built for the HLA RPR FOM. The *./bin5* directory contains executables for DIS versions 2.0.4, IEEE 1278.1-1995, and 2.1.4. The *./bin3* directory contains executables for DIS 2.0.3.

The *include* and *lib* Directories

The *./include* and *./lib* directories contain header files and libraries for *libLgrFile* which provides an API to the MÄK Logger file format. You need MÄK's VR-Link toolkit to use this API.

The *doc* Directory

The *./doc* directory contains the latest documentation in the Adobe Portable Document Format (PDF). With the free Acrobat Reader software, you can view, search, and print the documentation, and navigate using hypertext links. The reader is available for the PC and several UNIX platforms. To obtain a copy, visit www.adobe.com.

The *logger_tapes* Directory

The *./logger_tapes* directory contains recordings of two exercises. One exercise is provided in DIS 2.0.3 and 2.0.4 versions; the other in DIS 2.0.4 and HLA (RPR FOM) versions. For more information about these recordings, please see section 3.1.3, "Sample Logger Tapes".

2.3. Setting Up and Using the FLEXlm License Manager

The Logger, like other MÄK products, uses the FLEXlm license manager. Before you can use the Logger, you must have a license server installed and a valid *license.dat* file. (For information about FLEXlm, view the FAQ page at www.globetrotter.com/flmfaq.htm.) Contact the MÄK Technologies sales department for information about special licensing agreements.

2.3.1. Setting up the License Server

When you install the Logger, FLEXlm files are installed in the *./flexlm* directory. The procedure you need to follow to set up license management depends on whether or not you have previously installed a MÄK product that uses the license manager.

If you have no other MÄK products installed

The machine on which you run the license manager is the license server. It can be the same machine as the one on which you installed the Logger, or a different one. If you want to use a different machine as the license server, copy the files in the *./flexlm* directory to that machine.

To set up the FLEXlm license server:

1. Change to the *flexlm* directory on the license server.
2. Execute *lmhostid.* This program generates a unique number that MÄK uses to generate your license activation codes.
3. Call or e-mail MÄK with the name of the server and the number obtained by *lmhostid*. MÄK will e-mail or FTP you a file named *license.dat*.
4. Put the *license.dat* file in the *flexlm* directory on the license server and on all machines on which you have the Logger installed.
5. On each machine that has a *license.dat* file, set the environment variable LM_LICENSE_FILE to the absolute path to *license.dat*.

   On Windows 95, set environment variables in the *Autoexec.bat* file, for example:

   ```
   set LM_LICENSE_FILE="C:\Program Files\MÄK Technologies\
   flexlm\license.dat"
   ```

Note: Do not put spaces around the = sign. Put quotes around the path.

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

On UNIX, set the variable similarly to the following example:

```
setenv LM_LICENSE_FILE /usr/local/logger/flexlm/license.dat
```

6. Edit the *license.dat* file to specify the location of the *maklmgrd* executable on the server machine. The file supplied by MÄK represents this location with a dot as shown below:

   ```
   DAEMON maklmgrd .
   ```

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

   ```
   DAEMON maklmgrd "C:\Program Files\MÄK Technologies\
   flexlm\maklmgrd"
   ```

Note: On PCs, you must enclose the path in quotes.

If you already have a license.dat file from other MÄK products

If you have other MÄK products, you can set up a separate license server for the Logger, in which case follow the procedure on the previous page. Or, you can merge the license information for the Logger with that of previous products and use the same license server.

To obtain and merge license information:

1. Change to the *flexlm* directory on the license server.
2. Execute *lmhostid.* This program generates a unique number that MÄK uses to generate your license activation codes.
3. Call or e-mail MÄK with the name of the server, the number obtained by *lmhostid*. MÄK will e-mail or FTP you a file named *license.dat*. The file is in a format similar to the following:

```
SERVER mollusc 006097752f93 4567
DAEMON maklmgrd .
PACKAGE logger maklmgrd 1999.177 60D00041FA200D8A795D \
      COMPONENTS=logger1
INCREMENT logger maklmgrd 1999.177 1-jan-0 1 0BB480511615C175B8CF \
      PLATFORMS=i86_n
```

Your existing license.dat file is similar to the following:

```
SERVER mollusc 006097752f93 4567
DAEMON maklmgrd "c:\Program Files\MÄK Technologies\stealth\flexlm\
      maklmgrd"
PACKAGE makproduct maklmgrd 1999.177 40C020D12D609AB04D55 \
      COMPONENTS="hprod1 dprod1 prod1"
INCREMENT makproduct maklmgrd 1999.177 1-jan-0 1 AB4450916DD1944435D2 \
      PLATFORMS=i86_n
```

Note: The line continuation marks (\) in the examples are for clarity of presentation. They may not be present in the files you receive.

4. Open up the two files in a text editor. Append everything except the Server and Daemon lines (the Server lines will be identical) from the new *license.dat* file to the end of the old *license.dat* file. For example:

```
SERVER mollusc 006097752f93 4567
DAEMON maklmgrd “c:\Program Files\MÄK Technologies\stealth\flexlm\
     maklmgrd”
PACKAGE makproduct maklmgrd 1999.177 40C020D12D609AB04D55 \
     COMPONENTS="hprod1 dprod1 prod1"
INCREMENT makproduct maklmgrd 1999.177 1-jan-0 1 AB4450916DD1944435D2 \
     PLATFORMS=i86_n
PACKAGE logger maklmgrd 1999.177 60D00041FA200D8A795D \
     COMPONENTS=logger1
INCREMENT logger maklmgrd 1999.177 1-jan-0 1 0BB480511615C175B8CF \
     PLATFORMS=i86_n
```

5. Put the updated *license.dat* file in the *flexlm* directory on the license server and on all machines on which you have the Logger installed.

2.3.2. Running the License Server

The FLEXlm license server must be running on the server machine and the LM_LICENSE_FILE variable must be set on the local machine before you can run your MÄK applications.

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

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

Shutting Down the License Server

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

2.3.3. Troubleshooting the FLEXlm License Manager

For general troubleshooting information, refer to the FLEXlm FAQ at www.globetrotter.com/flmfaq.htm. It describes common problems, including the majority that have been reported to MÄK support engineers. Please consult the FAQ first when you have a problem.

Unable to Get A License - Port in Use

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

1. See if more than one *lmgrd* or *maklmgrd* process is running. On Windows NT, check the Task Manager. On UNIX, enter the following command:

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

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

2. If more than one of either process is running, note the process ids (PID) for both executables and kill all of them, for example:

   ```
   kill -9 1385 878
   ```

   where 1385 and 878 are process IDs.

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

3. See if there are old processes running that may be using a license. On Windows NT, check the task manager. On UNIX, use `ps -aux`.

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

4. If there are old processes present that may be using a license, kill them.
5. If in step 2 you killed the license server, start it with the `runLm` command.

Preventing Multiple License Manager Processes

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

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

2.4. Obtaining and Installing a Runtime Infrastructure (RTI)

An RTI (Run-Time Infrastructure) is a software library (and perhaps supporting executables) that implements the HLA Interface Specification. In HLA, applications exchange FOM data through RTI calls, which means that all HLA applications need to use an RTI. (Applications that use VR-Link typically do not need to make RTI calls directly, but only because VR-Link makes the RTI calls for them.)

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

The two RTI implementations against which all MÄK products have been tested are the MÄK RTI (which comes with all MÄK products), and the DMSO RTI (which is available from the DMSO web site).

2.4.1. Installing the MÄK RTI

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

UNIX

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

PCs

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

Configuring Your System to Use the MÄK RTI

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

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

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

- The FED file (*.fed*) for your federation execution (required)
- The RID file (*rid.mtl*) (optional)

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

Running Application with the MÄK RTI

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

Documentation for the MÄK RTI

The documentation for the MÄK RTI consists of:

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

2.4.2. Installing the DMSO RTI

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

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

Install the RTI according to the instructions provided.

Before you use the DMSO RTI for the first time, run the *configRTI* scripts provided in the root directory of any MÄK product. Provide the location of the RTI directory as an argument, as in these examples:

UNIX

```
configRTI /usr/RTI1.3
```

PCs

```
configRTI "c:\program files\rti\rti1.3"
```

configRTI copies the *VR-Link.fed* file from the MÄK products root directory to the DMSO RTI's config directory.

Configuring Your System to Use the DMSO RTI

To configure the environment:

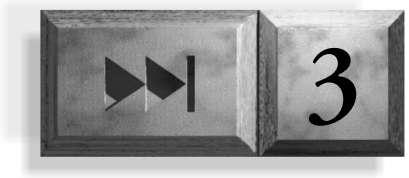
> On UNIX systems, source the *RTI.env* file.

> On Windows systems, set environment variables and add the RTI location to your path. The RTI's README file contains detailed instructions.

Running Applications Using the DMSO RTI

To use the DMSO RTI, you need to run the RTI executive (*rtiexec*). This executable resides in the RTI's *bin/platform_name* directory (UNIX) or in the RTI's *bin* directory (PCs). Run *rtiexec* only on one machine, regardless of how many HLA applications you are running.

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



Logger Concepts

This chapter provides an overview of what the Logger does, and describes features common to both Logger interfaces. For step-by-step instructions about how to operate the Logger, see Chapter 4, “Using The Logger Window”and Chapter 5, “Using The TLogger”.

This chapter covers the following topics:

3.1. Overview

The Logger is a data recorder that records HLA and DIS **exercises** and plays them 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 **recording**.

HLA only

The HLA version of the Logger performs recording and playback while acting as a separate federate. In record mode, 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 playback 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.

The Logger uses the same object IDs as those used by the federates when the **federation execution** was recorded.

In versions of the Logger prior to version 3.2, the Logger stores class, attribute, and parameter handles rather than names, so you need to be sure to use an identical FED file during recording and playback. In Logger 3.2 or later, Logger does not store handles, so you do not need to use identical FED files, but it is recommended.

3.1.1. Logger Interfaces

The Logger provides the following interfaces:

- The **Logger** window, a graphical user interface that uses VCR-style buttons to control the recording and playback of exercises.
- The **TLogger**, which uses a command line interface.
- An API which lets you control the Logger programmatically or over a network.

These interfaces have nearly identical capabilities. They let you:

- Start, stop, or pause when playing a recording.
- Navigate to a specific time-point when playing a recording.
- Control playback speed and direction.
- Select the network addresses used for the sending and receiving of data.

The only important functional differences between the graphical and character-based Loggers are:

- You can use a script to control the TLogger.
- The Logger window lets you change various options while the program is running. The TLogger requires that you set these options at startup through command-line switches.

3.1.2. Differences Between UNIX and Windows Versions

There are a few minor differences between the UNIX and Windows versions of the Logger.

Under UNIX, the entire Logger command line is case-sensitive, while under Windows, only the option letters in command-line options are case-sensitive.

For consistency across platforms, the hyphen is always used to precede command-line arguments, rather than the slash character typically used with DOS command-line arguments.

3.1.3. Sample Logger Tapes

The Logger includes sample recordings of two exercises:

- An exercise from the December 2, 1993 I/ITSEC show, Orlando, Florida. Two versions of this exercise are included with the Logger:

  `iitsec93.tape2` — contains DIS 2.0.3 data
  `iitsec93.dis204` — contains DIS 2.0.4 data.

  This exercise has been included with previous Logger releases.

> To view the exercise with the MÄK Stealth, run the *run-demo-low* script included with the Stealth.

- An exercise prepared for the MÄK/MultiGen/SGI demonstration at the 1995 I/ITSEC show:

  montyDemo.lgr — contains DIS 2.0.4 data.
  montyDemoHLA.lgr — contains HLA data.

  This recording is of an exercise in which two A-10s attempt to destroy seven ground targets near an airfield.

> To view this exercise with the MÄK Stealth, run one of the *run-demo-hi* scripts included with the Stealth.

Unlike most Logger recordings, the *montyDemo* recording contains View Control PDUs, which cause the Stealth to follow the action automatically, switching among many predetermined viewpoints. For information about manually controlling the eyepoint in this demo, see your Stealth documentation.

3.2. What the Logger Does During Recording and Playback

This section describes, in general, what happens when you record and play back exercises. When you start the Logger, you don't need to specify record or playback **mode**. You can do this in the interface. When you start the TLogger, you must specify the mode. For complete instructions for starting the Logger, see section 4.1.1, "Starting the Logger". For complete instructions for starting the TLogger, see "Recording Exercises", on page 5-2.

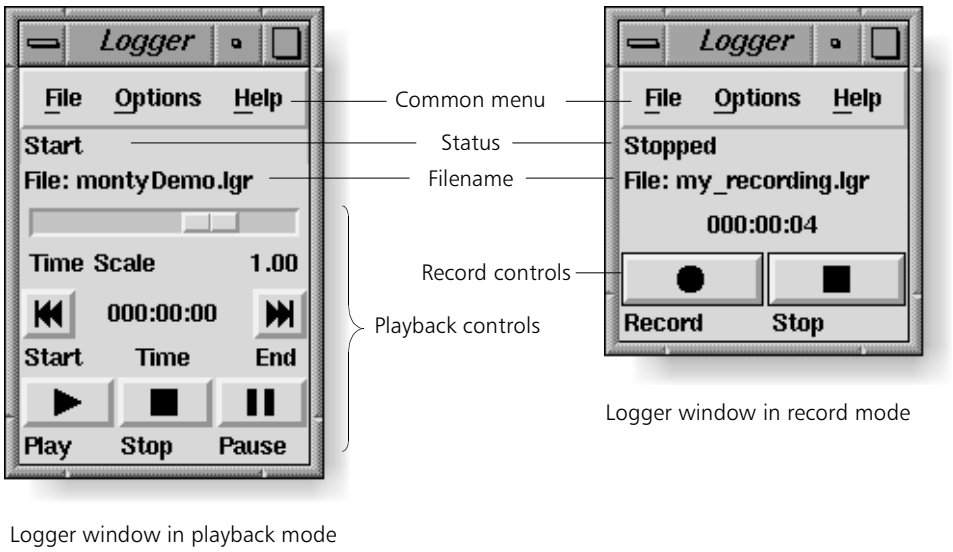


Figure 3-1. The Logger windows

3.2.1. What The Logger Does During Recording

In record mode, the Logger is in either a recording state or a stopped state. When stopped, the Logger does not record data and does not advance time. When running the TLogger, you can select a recording or stopped state with the `go` and `stop` runtime commands. In the Logger, use the Record and Stop buttons to control the **state**.

DIS only

In record mode, the DIS Logger records all DIS packets on UDP port 3000, without regard to exercise ID. You can specify a different UDP port at startup by using the `-P` switch, for example:

```
Logger -r newfile.lgr -P 2727
```

With each PDU recorded, the Logger stores a timestamp indicating when the PDU occurred relative to the start of the recording.

HLA only

Under HLA, the Logger subscribes to objects, attributes, and interactions, and 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.

3.2.2. What The Logger Does During Playback

The Logger may or may not begin playback immediately on startup, depending on which additional command-line switches you use.

When operating in playback mode, the Logger is in either a playing state, a paused state, or a stopped state. When playing, the Logger reads messages from the recorded file and transmits them in sequence over the network. When paused, the Logger halts at the current timepoint in the file, and in DIS, repeatedly retransmits messages that represent the current state of each entity to maintain the DIS **heartbeat** (This retransmission is not needed in HLA). When stopped, the Logger halts at the current timepoint in the file and stops sending data.

In the TLogger, you can select a playing, paused, or stopped state with the `go`, `pause`, and `stop` commands. In the Logger, use the Play, Pause, and Stop buttons to control the state.

During playback, you can control the playback speed and direction to produce fast motion, slow-motion, and reverse playback effects. See section 3.3, "Timescale and Direction During Playback" and section 3.4, "Compensating for Timescale Effects During Playback" for details.

Logger lets you navigate to a specified timepoint in a recording, or skip forward or backward by a specified time increment. You navigate time using the Start and Stop buttons, and the Skip Time and Set Time menu options (described in section 4.4, "Navigating to A Specific Time"). In the TLogger, you use the `start`, `end`, `time`, and `skip` commands (see section 5.3, "Navigating Time").

DIS only

In DIS, when you start the Logger and specify a file for playback, the Logger sends the recorded PDUs to UDP port 3000. You can use the `-P` switch to specify a different port, for example:

```
Logger -p demofile.lgr -P 2727
```

HLA only

When playing back in HLA, the Logger acts as a federate that owns each of the objects represented in the playback 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.

3.3. Timescale and Direction During Playback

The Logger lets you play a recording at any speed within the capabilities of your platform, forward or backward.

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 forward, half-speed playback. A negative timescale causes reverse playback. Figure 3-2 shows a set of typical timescale values and their effect on speed and direction.

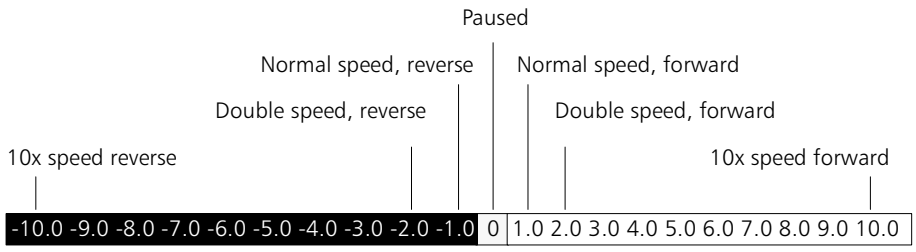


Figure 3-2. Sample timescale values

In the Logger, you use a slider control to set the timescale or enter the timescale value as a numeric value (see section 4.5, "Setting Timescale and Direction"). In the TLogger, you control timescale with the `scale` command (see section 5.2, "Playing Back Recordings").

You can specify an initial timescale at Logger startup with the `-s` option, for example:

```
Logger -p my-exercise.lgr -s 2.5
```

3.4. Compensating for Timescale Effects During Playback

The Logger has corrective features that adjust the content of outgoing messages for non-normal playback speeds. Under DIS, Interim PDU generation prevents 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 non-standard playback speed.

3.4.1. Generating Interim PDUs During Slow-Speed DIS Playback

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 five seconds.

Most receiving applications typically 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 time the entity out. A typical timeout threshold is 12 seconds, or 2.4 times the heartbeat rate.

Using Interim PDUs to Maintain Heartbeat

During slow-speed playback, 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 DIS Logger calculates the contents of an interim PDU by applying a dead-reckoning algorithm to an entity's most recent PDU.

By default, the DIS Logger ensures that at least one PDU is transmitted for each entity every five seconds. You can specify a different heartbeat rate. In the Logger, the heartbeat setting is in the Configuration dialog box, available through the Options menu (see section 4.6, "Setting Configuration Options".) In the TLogger, use the `heartbeat` run-time command (see section 5.6, "Controlling DIS Heartbeat and Timeout".)

Scaling the Timeout Period During Slow-Speed Playback

When an entity's PDUs stop occurring in the recording, the DIS 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 recording at half-speed, using the default timeout setting of 12 seconds, the Logger will generate interim PDUs for an entity for 12 recorded seconds, or 24 real-time seconds before timing it out.

To set a different timeout threshold in the Logger, change the timeout setting in the Configuration dialog box, available through the Options menu. In the TLogger, use the `timeout` run-time command. See section 4.6, "Setting Configuration Options" and section 5.6, "Controlling DIS Heartbeat and Timeout" for details.

3.4.2. Updating Entity Information Quickly After a Time Jump

During playback, the you can jump to a specified timepoint using the Set Time or Skip Time option, or the `time` or `skip` command in the TLogger. 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.

To speed the update process, the Logger provides a feature called **entity maintenance**. 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.

In the Logger window, entity maintenance is applied when you issue a Skip Time command, but not for a Set Time command.

In the TLogger, you can turn entity maintenance on or off separately for the `time` and `skip` commands with the `timemaint` and `skipmaint` commands. See section 5.4, "Controlling Entity Maintenance" for details.

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

3.4.3. Velocity and Acceleration Scaling

When playing at speeds above or below normal, the Logger can adjust the velocity and acceleration values in entity state messages accordingly. 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 be move at double its original speed when viewed using MÄK 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.

This scaling feature is active by default. To disable it, use the `-d` command-line option at Logger startup.

DIS only

Scaling must be enabled in order for the DIS Logger to send PDUs while in the paused state. If a file contains bundled PDUs, the Logger does not unbundle them on playback, and therefore cannot scale their speed and velocity.

3.4.4. Effect of Reverse Playback on Dead Reckoning

Applications receive state update messages in reverse order during reverse playback. Because dead-reckoning is based on acceleration, and acceleration is not a mathematically reversible operation, your applications may place a vehicle on a path different from the one originally traveled when receiving a reverse playback.

3.5. The End-of-File Action

When the Logger reaches the end of a file, it can enter a stopped or paused state, or it can wrap to the other end of the file and continue playback.

> To specify an end-of-file action, start the Logger with the `-e` command-line option:

```
Logger -es     Stop (default)
Logger -ep     Pause
Logger -ew     Wrap
```

In the Logger window, you can also change the end-of-file action in the Configuration dialog box, available through the Options menu. See "Setting Configuration Options", on page 4-9.

During reverse playback, if wrap is in effect and the beginning of the tape is reached, the tape wraps to the end and continues reverse playback.

3.6. DIS-Specific Settings

In DIS, the Logger can direct its output to a specific destination address during playback, and can listen (subscribe) to one or more multicast addresses during recording. You can also control which UDP port is used for playback or recording.

3.6.1. Specifying a Destination Address for Outgoing Data

Normally, the Logger sends outgoing data packets to the host machine's default broadcast address.

> To specify a different destination address, use the `-A` command-line option, as in this example:

```
Logger -p my_exercise.lgr -A 123.234.123.234
```

The specified destination address may be a broadcast, multicast, or unicast address.

3.6.2. Subscribing to Multicast Groups

In record mode, the Logger can subscribe to multicast groups through use of the `-S` command-line option. When subscribed to multicast groups, the Logger also continues to receive data through the machine's broadcast and unicast addresses.

> To subscribe to more than one multicast address, use the `-S` option several times on the same command line:

```
Logger -r newfile.lgr -S 225.0.0.118 -S 225.0.0.202
```

Subscription pertains to record mode only.

Multicasting is fully supported under IRIX-5, IRIX-4, and Solaris-2. Under DEC OSF/1 v2.0, applications can receive, but not send, multicast.

Note: Multicast subscription is a property of the host machine, not of individual applications. Therefore, 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. Accordingly, if the Logger is running on the same machine and using the same port as other applications that subscribe to multicast groups, the Logger may receive traffic intended for the other applications.

3.6.3. Specifying a UDP Port

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

> You can specify the UDP port start the Logger with the uppercase `-P` switch, as in these examples:

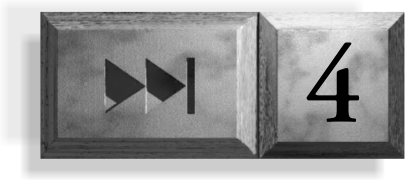
```
Logger -P 2727
Logger -p mydemo.lgr -P 2727
```

3.7. HLA-Specific Settings

By default, the HLA Logger uses the federation execution name VR-Link.

> To specify a federation execution name, use the `-x` option at startup. Replace *ExName* with the desired name:

```
Logger -p my_exercise.lgr -x ExName
```



Using The Logger Window

This chapter covers the following topics as they apply to the Logger's graphical interface:

4.1. The Logger Windows

The Logger window has separate control sets for playback and record modes. In playback mode, the Logger has a slider for scaling and reversing time, and a set of time navigation controls. In record mode, the Logger only has recording controls. Figure 4-1 illustrates the Logger window.

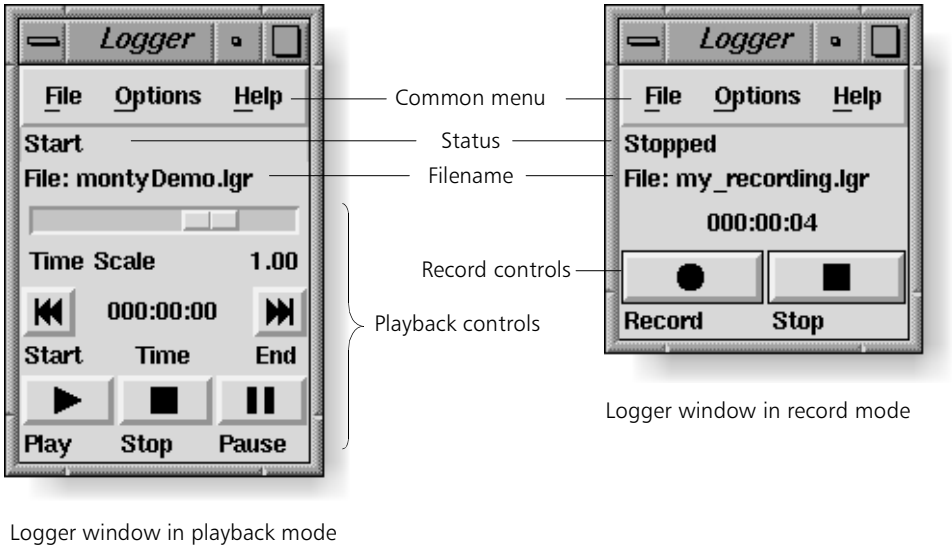


Figure 4-1. The Logger window

4.1.1. Starting the Logger

You can use command-line options to set many of the Logger's initial settings. However, because the Logger window provides run-time access to most frequently-used options, you can usually start the Logger without options.

Table 4-2 lists startup options for the Logger.

Starting the Logger in UNIX

To start the Logger:

1. Open a UNIX shell.
2. Change to the appropriate */bin* directory.
3. Enter:

   ```
   Logger [options]
   ```

Starting the Logger in Windows

Start Logger in any of the following ways:

> In a DOS console window, switch to the appropriate *\bin* directory, and enter:

```
Logger [options]
```

When the Logger window and its companion console window appear, you can close the initial DOS window if you want to.

> Choose the icon from the Start Menu or folder.

> From the Start menu, choose Run. Enter:

```
Logger [options]
```

Note: Some options (such as the federation execution name in HLA, or the DIS port or destination address) are not available through the Logger window and therefore must be specified on the command line at startup.

To invoke command-line options when starting the Logger from an icon, add the desired options to the startup command line in the icon's property sheet. You can create several Logger shortcuts (or icons), each with a different set of command-line options. See your Windows documentation for more information.

4.2. Recording Exercises

To begin recording:

1. In the Logger window, choose File→Record New File. The Select File dialog box opens (Figure 4-2.)
2. If necessary, navigate to the proper directory, then enter a new filename or select an existing file.

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

3. Click OK.

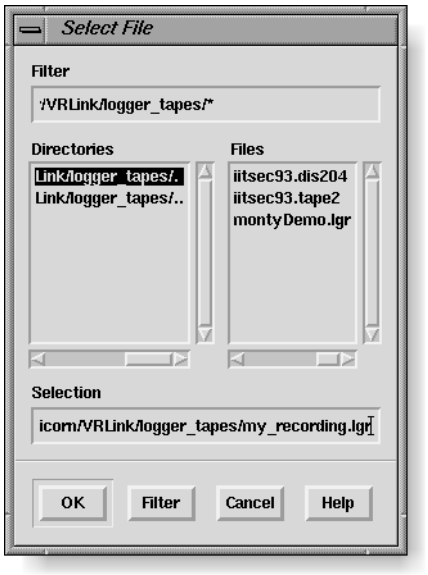


Figure 4-2. Specifying a destination file for recording

The Logger enters record mode in a stopped state. To begin recording or return to a stopped state, click the appropriate button. The Status field displays either Recording or Stopped accordingly.

As recording progresses, the Time field is updated to display elapsed time. In record mode, the timescale is fixed at 1.0 (normal speed).

DIS only

In DIS, the Logger records all traffic that the host machine receives on the default UDP port (3000). If desired, you can use startup command-line options to specify a different UDP port or subscribe to additional multicast addresses. For more information about these options, see section 3.6, “DIS-Specific Settings”.

HLA only

In HLA, the Logger records all updates, interactions and object removals that it perceives through the RTI.

4.3. Playing Back Exercises

To play back an existing exercise:

1. In the Logger window, choose File→Play Back Old File. The Select File dialog box opens (Figure 4-2.)
2. If necessary, navigate to the desired directory, then select the file to be played.
3. Click OK.

By default, the Logger enters a stopped state when the file is loaded. To play, pause, or stop playback, click on the appropriate button.

If the recording is at its beginning or end, the status field displays Start or End. Otherwise, Stopped, Paused, or Playing is displayed to reflect the Logger's current operating state.

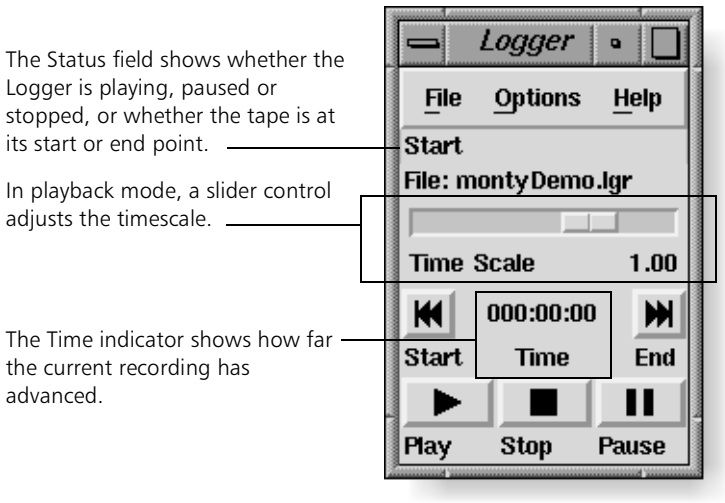


Figure 4-3. Logger playback controls

DIS only

In DIS, the Logger sends data to host machine's default broadcast address over the default UDP port (3000). If desired, you can use command-line options to specify a different UDP port or destination IP address. See section 3.6, "DIS-Specific Settings" for more information. The Logger transmits PDUs while in the playing or paused states. In the stopped state, no PDUs are sent.

4.4. Navigating to A Specific Time

The Logger provides the following ways to navigate to a specific time in the exercise:

- Use the Start and End buttons in the Logger's main window to jump to the beginning or end of the recording.
- Use the Skip Time option to jump forward or backward in time by a specified offset.
- Use the Set Time option to jump to a specific time.

The Skip Time and Set Time options are on the Options menu.

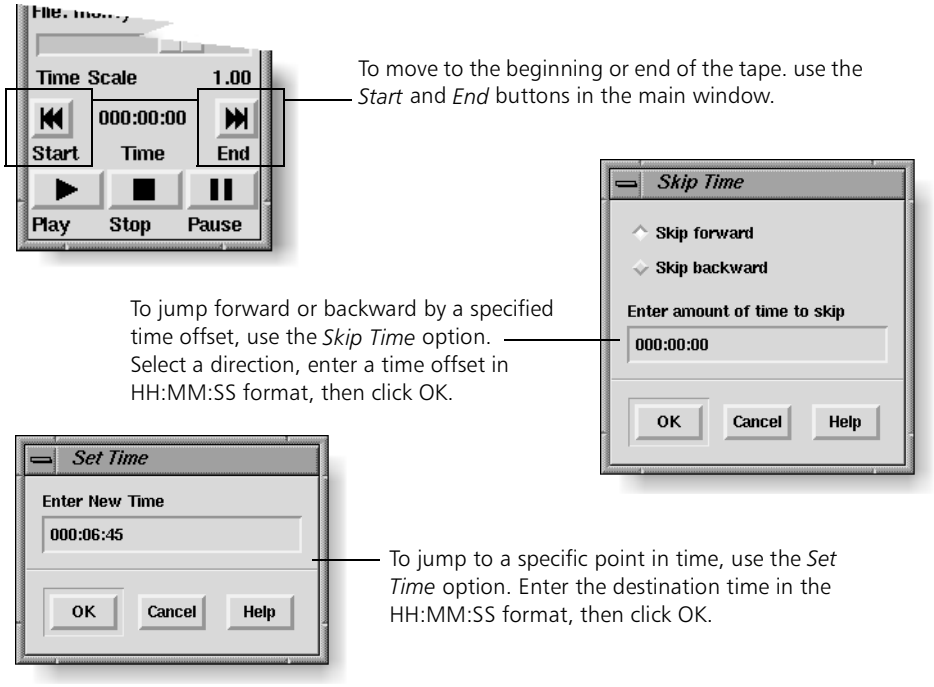


Figure 4-4. Time Navigation in the Logger

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. For more information, see section 3.5, "The End-of-File Action", and section 4.6, "Setting Configuration Options".

During slow-speed playback, when you use the Skip Time option, the Logger performs entity maintenance to speed the transmission of up-to-date messages. When you use the *Set Time* option, the Logger does not perform entity maintenance. For more information, see "Updating Entity Information Quickly After a Time Jump", on page 3-8.

4.5. Setting Timescale and Direction

The Logger lets you play a recording forward or backward at any speed, limited only by the capabilities of your hardware. You can set the timescale in the following ways:

- Adjust a slider control in the Logger main window. The slider can be customized to contain a specific set of timescale values.
- Specify an initial timescale at Logger startup through a command-line option.
- Enter the desired timescale into a dialog box.

These options are described in the following sections. For information about timescale values, see section 3.3, "Timescale and Direction During Playback".

4.5.1. Using and Customizing the Timescale Slider

The Timescale slider control in the Logger main window is the most convenient way to change playback speed and direction. In its default configuration, the slider contains nineteen preset values ranging from -10 to +10.

> To change the timescale, drag the slider button.

If you need finer control, click once on the slider button to give it focus, then use the left and right arrows on your keyboard to step through the values.

> To specify the values that appear on the timescale slider, start the Logger, with the `-t` command-line option followed by a comma-separated list of values, as shown below. Do not include any spaces in the value list:

```
Logger -t-5,-4,-3,-2.5,-2,-1.5,-1,0,1,1.5,2,2.5,5,7
```

The values appear in ascending order on the slider control, even if you enter them out of sequence.

4.5.2. Setting a Timescale at Startup

> To specify a timescale at Logger startup, use the `-s` command-line option, as in this example:

```
Logger -p my-exercise.lgr -s 2.5
```

4.5.3. Entering A Timescale Value Numerically

If the timescale you want is not available through the slider control, you can specify it directly.

1. In the Logger window, choose Options→Set Timescale. The Set Time Scale dialog box opens (Figure 4-5.)

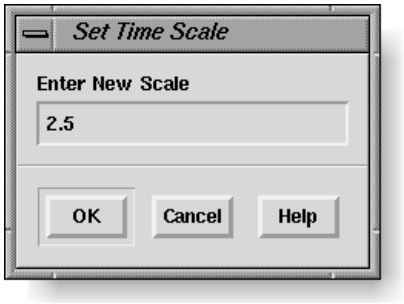


Figure 4-5. Set Time Scale dialog box

2. Enter the desired timescale.
3. Click OK.

You can enter the timescale value in scientific notation, for example, `1e+4` instead of `10,000`.

Note: Although the Logger lets you specify a very large timescale (beyond 1e+35), the use of an unusually large timescale may produce unexpected results because of hardware limitations.

If a recording is played at a very high speed, the Logger may be unable to keep up, especially during passages that contain a high concentration of data. If the Logger falls significantly behind, it displays the Timescale value in red and continues to process the messages as quickly as possible. None are discarded.

4.5.4. Message Processing for Non-Standard Timescales

When playing at speeds above or below normal, the Logger makes corrective adjustments to the velocity and acceleration values in outgoing entity state messages. For details, see section 3.4.3, “Velocity and Acceleration Scaling”.

DIS only

In DIS, the Logger can generate interim PDUs during slow-speed playback to maintain heartbeat and prevent receiving applications from timing entities out too soon. For more information, see section 3.4.1, “Generating Interim PDUs During Slow-Speed DIS Playback”, and section 4.6, “Setting Configuration Options”.

4.6. Setting Configuration Options

To change configuration options:

1. Choose Options→Configuration. The Configuration dialog box opens (Figure 4-6.)

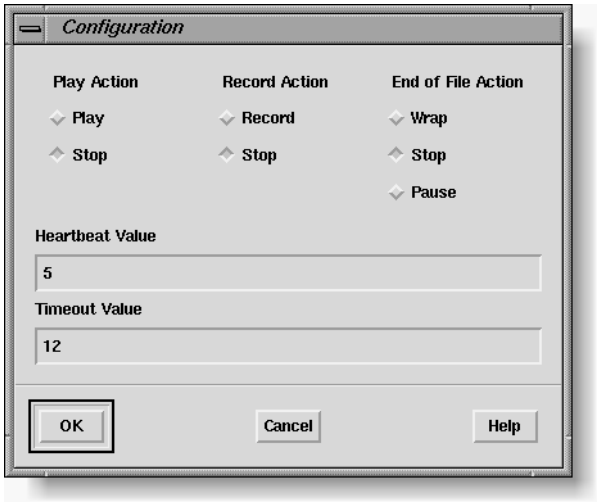


Figure 4-6. The Configuration dialog box

Table 4-1 describes the configuration options.

Table 4-1: Logger Configuration Options (Sheet 1 of 2)

Option	Description
Play Action	Determines whether the Logger enters a stopped state (default) or begins playing immediately when a file is opened for playback. To set this option at startup, use the `-os` or `-op` option.
Record Action	Determines whether the Logger enters a stopped state (default) or begins recording immediately when a file is opened for recording. To set this option at startup, use the `-bs` or `-br` option.
End of File Action	When the end of a Logger tape is reached during playback, this option determines whether the Logger stops, pauses, or wraps around to the other end of the tape to continue playback (stop is the default). To set the end-of-file action at startup, use the `-es`, `-ep`, or `-ew` option. See section 3.5, "The End-of-File Action"for more information.
Heartbeat (DIS only)	During slow-speed playback in DIS, this option determines how frequently (in seconds) the Logger generates interim PDUs for each entity. For information, see section 3.4.1, "Generating Interim PDUs During Slow-Speed DIS Playback".

Table 4-1: Logger Configuration Options (Sheet 2 of 2)

Option	Description
Timeout (DIS only)	During slow-speed playback in DIS, this option determines how long the Logger continues to generate interim PDUs for an entity once that entity's PDUs have stopped occurring in the recording. For more information, see section 3.4.1, "Generating Interim PDUs During Slow-Speed DIS Playback".

Note: Configuration changes are valid only for a given Logger session. They are not retained for future sessions.

4.7. Summary of Startup Options for the Logger

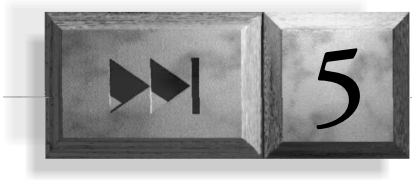
Table 4-2 lists command line options. They are grouped by mode and indicate if they are for HLA only or DIS only.

Table 4-2: Command-line options (Sheet 1 of 2)

Option	Description
	Record and Playback
`-h, -help, -?`	Displays a brief summary of Logger command-line options.
`-P` *`dis_port`*	DIS only. Specifies which DIS port (UDP port) the Logger uses to send or receive data. The default port is 3000.
`-x` *`ExName`*	HLA only. Sets federation execution name. VR-Link is default.
`-a "0:0:0"`	DIS only. Specifies the site:host:entity for remote control of the Logger.
`-v {0|1}`	Specifies visibility of the Logger interface for remote control. 1 = Logger interface visible (default) 0 = Logger interface not visible
`-L {0|1|2}`	Specifies the response to Logger Control PDUs for remote control. 0 = Do not respond to Logger Control PDUs 1 = interpret Logger Control PDUs only if they are addressed to the entity ID specified by the -a argument (default) 2 = interpret all Logger Control PDUs
	Record Mode Only
`-b` *`state`*	Specifies whether the Logger enters a stopped state or begins recording immediately when opening a file for recording. Set *`state`* to `r` for record, or `s` for stopped (default).
`-r [`*`filename`*`]`	Causes the Logger to open in record mode; specifies the destination file for recorded data. With the Logger, the `-r` switch is optional, and may be used without a filename. (You choose the filename from the Logger window.)
`-R {0|1}`	Specifies whether or not to record Logger Control PDUs during remote control. 0 = do not record Logger Control PDUs (default) 1 = record Logger Control PDUs

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

Option	Description
-S *mcastAddr*	DIS only. Subscribes the Logger to the multicast address specified by *mcastAddr*. To subscribe to several multicast addresses, use this option more than once on the command line. See section 3.6.2, "Subscribing to Multicast Groups".
Playback Mode Only	
-A *addr*	DIS only. Sets the default IP address to which the Logger sends PDUs. If this option is not used, the Logger uses the computer's broadcast address. See section 3.6.1, "Specifying a Destination Address for Outgoing Data".
-d	Disables the generation of velocity and acceleration scaling data when playing at speeds above or below normal. See section 3.4.3, "Velocity and Acceleration Scaling".
-e *action*	Specifies the Logger's action upon reaching the end of the file. Set *action* to p to pause at the file's end, w to wrap to the other end of the tape and continue, or s to stop (default).
-o *state*	When a file is opened for playback, specifies whether the Logger enters a stopped state, or begins playing immediately. Set *state* to p for playing, or s for stopped (default).
-p [*filename*]	Causes the Logger to open in playback mode; specifies the source file to be played. With the Logger, the -p switch is optional, and may be used without a filename. (You choose the filename from the Logger window.)
-s *timescale*	Sets the initial playback timescale factor. For more information, see section 3.3, "Timescale and Direction During Playback".
-T *timepoint*	Sets the timepoint in the file where playback begins. Enter the *timepoint* in HH:MM:SS format. Default is 00:00:00
-t *list*	Specifies the list of values that appear on the Logger timescale slider control. Enter *list* as a series of comma-separated values without spaces. See section 4.5.1, "Using and Customizing the Timescale Slider".



Using The TLogger

The TLogger offers the same features as the graphical interface, plus script-based recording and playback capability. This chapter covers the following topics:

5.1. Recording Exercises

To begin recording:

1. Open a console window.
2. Enter the following command:

   `TLogger -r exercise_name.lgr`

   where *`exercise_name.lgr`* is the name of the file to which you want to record this exercise.

At startup, a status message shows the current settings. The following example is from a DIS exercise. It shows port, broadcast address, destination file, and current operating status.

```
recordfile my-exercise.lgr
NetSocket ok for port 3000 to 123.456.789.255
Status: Select File
Status: Recording
Logger file my-exercise.lgr is version 2 (writing)
my-exercise.lgr
Cannot adjust timescale while recording.
Type "i" for information, "?" for help.
The Current Status is : Recording
```

DIS only

The TLogger records all DIS traffic that the host machine receives on UDP port 3000. You can use command-line options to specify a different UDP port or to subscribe to multicast groups. See section 3.6, “DIS-Specific Settings” for more information.

HLA only

The TLogger records all updates, interactions, and object removals that it perceives through the RTI.

5.1.1. TLogger Run-time Commands for Recording

Recording with the TLogger is straightforward. At the TLogger prompt, enter:

`go`	To begin or resume recording.
`stop`	To stop recording without ending the session.
`quit`	To close the file and exit the TLogger.

5.1.2. Displaying Recording Status

At any time during the recording, you can enter the `info` command (or the letter `i`) to display a brief status summary. The summary shows the destination file name, the time elapsed since the start of recording, and the current operating status.

```
> info
FILE: my-exercise.lgr
TIME: 000:01:43
SCALE: 1.000000
STATE: Recording
>
```

5.1.3. Beginning a Recording Session in a Stopped State

By default, when you start the TLogger in record mode, it begins recording immediately.

> To begin in a stopped state, use the `-b` option to set the record action as in this example:

```
TLogger -bs -r my-exercise.lgr
```

5.2. Playing Back Recordings

To play a Logger file:

1. Open a console window.
2. Enter the *TLogger* command with the `-p filename` argument, for example:

```
TLogger -p ../logger_tapes/my-exercise.lgr
```

DIS only

The *TLogger* command opens the specified file, and sends the recorded DIS information to the host machine's default broadcast address over UDP port 3000. You can use command-line options to specify a different UDP port or destination IP address. See section 3.6, "DIS-Specific Settings" for more information.

The DIS status message below shows the current port, destination IP address, source file, and current operating status.

```
playfile my-exercise.lgr
NetSocket ok for port 3000 to 123.456.789.255
Status: Select File
Status: Playing
Logger file my-exercise.lgr is version 2 (writing)
my-exercise.lgr
Type "i" for information, "?" for help.
The Current Status is : Playing
```

5.2.1. Basic TLogger Playback Commands

In playback mode, the TLogger starts in a playing state. Thereafter, it is in either a playing, paused, or stopped state. At the TLogger prompt, enter:

`go`	To begin or resume playback. When playing, the TLogger sends out PDUs in chronological sequence.
`pause`	To pause playback. When paused, the TLogger stops time and, for DIS only, repeatedly rebroadcasts each entity's most recent entity state PDU.
`stop`	To stop playback. When stopped, the TLogger stops time and does not transmit data.
`quit`	To close the file and exit the TLogger.

5.2.2. Displaying Playback Status

> To display a playback status summary, enter the `info` command (or the letter "`i`") at the TLogger prompt.

The summary shows the source file name, the current timepoint in the Logger file, the timescale (playback speed), and the current operating status.

```
> info
FILE: my-exercise.lgr
TIME: 000:00:09
SCALE: 1.000000
STATE: Playing
```

5.2.3. Beginning a Playback Session in a Stopped State

By default, when you start the TLogger in playback mode, it begins playback immediately.

> To begin in a stopped state, use the `-o` option to set the play action as in this example:

```
TLogger -os -p demo.lgr
```

5.3. Navigating Time

During playback, four commands let you navigate to a specific point in a recording:

`start` moves to the beginning of the recording.

`end` moves to the end of the recording.

`time` sets the timepoint to a specified time. The command format is:

```
time HH:MM:SS
```

`skip` causes the Logger to jump forward or backward in time by a specified offset. Enter the offset in an `HH:MM:SS` format, or as a number of seconds. Use a negative value to skip backward in time. For example:

```
skip 00:02:30
skip -170
```

If you execute a time jump while playback is paused or stopped, the TLogger retains that state. The TLogger also retains a playing state after a time jump unless the end of the file is reached and wrapping is disabled. For more information, see section 3.5, "The End-of-File Action", and section 5.3.1, "Controlling End-of-File Behavior".

When you use the `time` or `skip` command during slow-speed playback, the TLogger can perform entity maintenance to facilitate the transmission of up-to-date PDUs. For a description of entity maintenance, see "Updating Entity Information Quickly After a Time Jump", on page 3-8. For information about how to enable or disable entity maintenance, see section 5.4, "Controlling Entity Maintenance".

5.3.1. Controlling End-of-File Behavior

The TLogger normally enters a stopped state when it reaches the end of a tape.

> To specify the end-of-file action, start the Logger with the `-e` option, as follows:

```
TLogger -p filename -e [p|w]
```

where `p` causes the Logger to enter a paused state at the end of a file, and `w` causes the Logger to wrap around to the other end of the file and continue playback.

5.4. Controlling Entity Maintenance

When you use a `time` or `skip` command during slow-speed playback, the TLogger's entity maintenance feature can ensure that up-to-date entity state messages are transmitted as soon as possible after the time jump. Entity maintenance is described in detail in "Updating Entity Information Quickly After a Time Jump", on page 3-8.

By default, the TLogger performs entity maintenance when you issue a `skip` command, but not when you issue a `time` command.

> To change entity maintenance settings, issue one of the following commands:

```
timemaint n
skipmaint n
```

where *n* is 1 to enable entity maintenance, and `n` is 0 to disable it.

> To view the current `timemaint` or `skipmaint` setting, enter the command without a value at the TLogger prompt.

5.5. Setting Playback Timescale and Direction

You can play a recording at any practical speed, forward or backward.

> To set the playback speed and direction, enter the `scale` command:

```
scale timescale
```

where `timescale` is the factor by which time is to be multiplied. For information about choosing timescale values, see section 3.3, "Timescale and Direction During Playback".

> To specify a timescale at startup, use the `-s` option, for example:

```
TLogger -p my-exercise.lgr -s 2.5
```

You can enter the timescale value in scientific notation, as in the following example for 10,000:

```
scale 1e+4
```

Note: Although the TLogger lets you specify a very large timescale (beyond 1e+35), the use of an unusually large timescale may produce unexpected results because of hardware limitations, and is not recommended.

If a recording is being played at a high speed, the TLogger may be unable to keep up, especially during passages that contain an unusually high concentration of data. If the TLogger falls behind, it continues to process messages as quickly as possible. No messages are discarded.

5.5.1. Message Processing for Non-Standard Timescales

When playing at speeds above or below normal, the TLogger makes corrective adjustments to the velocity and acceleration values in outgoing state update messages. For details, see section 3.4.3, "Velocity and Acceleration Scaling".

DIS only

In DIS, the TLogger can generate interim PDUs during slow-speed playback to prevent receiving applications from timing entities out too soon. For more information, see section 3.4.1, "Generating Interim PDUs During Slow-Speed DIS Playback", and section 5.6, "Controlling DIS Heartbeat and Timeout".

5.6. Controlling DIS Heartbeat and Timeout

During slow-speed playback in DIS, the TLogger generates interim PDUs as needed to maintain an appropriate heartbeat rate and timeout threshold for each entity in the recording. Heartbeat and timeout concepts are explained in section 3.4.1, "Generating Interim PDUs During Slow-Speed DIS Playback".

5.6.1. Setting the Heartbeat Rate

The heartbeat rate determines how frequently the TLogger generates interim PDUs for each entity. By default, the TLogger maintains a heartbeat rate of five seconds.

> To set the heartbeat rate, use the `heartbeat` command:

```
heartbeat n
```

where *n* represents the maximum allowable time between PDUs.

> To view the current `heartbeat` setting, enter the command name without a value.

5.6.2. Setting the Timeout

The timeout threshold determines how long the TLogger continues to generate interim PDUs for an entity once the entity's PDUs stop occurring in the recording. The TLogger's default timeout setting is twelve seconds.

> To set the timeout value, use the *`timeout`* command:

```
timeout n
```

where *n* represents the number of recorded seconds the TLogger should wait before timing an entity out.

> To view the current `timeout` setting, enter the command name without a value.

5.7. Using a Script to Control Recording and Playback

A script is a plain text file that can control the TLogger in either playback or record mode. Scripts are especially useful for creating a prepared demo, or for automatically bypassing unwanted segments during tape playback.

A script can contain any TLogger run-time commands listed in Table 5-2. Create the script in any text editor, then redirect it to the Logger as in these examples:

```
TLogger -p my_exercise.lgr < my_play_script.txt
TLogger -r new_exercise.lgr < my_record_script.txt
```

At startup, the TLogger enters a playing or recording state by default, then immediately begins processing commands in the script.

Controlling Processing with the Wait Command

The `wait` command controls how long the TLogger runs in its current state before processing the next command in the script. The sample script below provides an example.

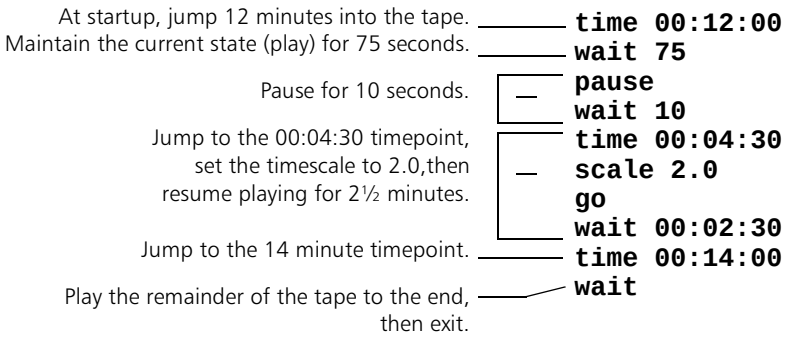


Figure 5-1. Sample TLogger script

If `wait`, with a *`time`* argument, is the last command in a script, the TLogger maintains the current state for the duration of time argument, then exits.

If a script contains a `wait` command with no *`time`* argument, the TLogger continues operating in the current state until reaching the end of the tape, then exits. Any subsequent commands in the script are ignored.

5.7.1. Stopping A Script

The Logger does not respond to keyboard input when running a script, so you cannot manually enter `quit` to stop execution.

> To stop a script, enter `Ctrl-C`.

5.8. Summary of TLogger Startup Options

Table 5-1 lists the parameters to the TLogger command.

Table 5-1: TLogger command-line options (Sheet 1 of 2)

Option	Description
	Playback and Record
`-h, -help, -?`	Displays a brief summary of TLogger command-line options.
`-m` *`char`*	Sets the TLogger's prompt character to *`char`*. Default is `>`.
`-P` *`dis_port`*	DIS only. Specifies which DIS port (UDP port) the TLogger uses to send or receive data. The default port is 3000.
`-x` *`ExName`*	HLA only. Sets the federation execution name. *VR-Link* is the default.
`-a "0:0:0"`	DIS only. Specifies the site:host:entity for remote control of the Logger.
`-v {0|1}`	Specifies visibility of the Logger interface for remote control. 1 = Logger interface visible (default) 0 = Logger interface not visible
`-L {0|1|2}`	Specifies the response to Logger Control PDUs for remote control. 0 = Do not respond to Logger Control PDUs 1 = interpret Logger Control PDUs only if they are addressed to the entity ID specified by the -a argument (default) 2 = interpret all Logger Control PDUs
	Record Mode Only
`-b` *`state`*	Specifies whether the TLogger opens in a stopped state or begins recording immediately. Set *`state`* to `r` (TLogger default) for record, or `s` for stopped.
`-r` *`filename`*	Causes the TLogger to open in record mode; specifies the destination file for recorded data. When recording with the TLogger, the -r switch and filename are required.
`-R {0|1}`	Specifies whether or not to record Logger Control PDUs during remote control. 0 = do not record Logger Control PDUs (default) 1 = record Logger Control PDUs
`-S` *`mcastAddr`*	DIS only. Subscribes the TLogger to the multicast address specified by *`mcastAddr`*. To subscribe to several multicast addresses, use this option more than once. For information, see section 3.6.2, "Subscribing to Multicast Groups".

Table 5-1: TLogger command-line options (Sheet 2 of 2)

Option	Description
Playback Mode Only	
-A *addr*	DIS only. Sets the default IP address to which the TLogger sends PDUs. If this option is not used, the TLogger uses the computer's broadcast address. See section 3.6.2, "Subscribing to Multicast Groups".
-d	Disables the generation of velocity and acceleration scaling data when playing at speeds above or below normal. See section 3.4.3, "Velocity and Acceleration Scaling".
-e *action*	Specifies the TLogger's action upon reaching the end of the file. Set *action* to p to pause at the file's end, w to wrap to the other end of the tape and continue, or s to stop (default).
-o *state*	Specifies whether the TLogger opens in a stopped state or begins playback immediately. Set *state* to p for playing, or s for stopped (TLogger default).
-p *filename*	Causes the TLogger to open in playback mode and specifies the source file to be played. When playing back with the TLogger, the -p switch and filename are required.
-s *timescale*	Sets the initial playback timescale factor. For more information, see section 3.3, "Timescale and Direction During Playback".
-T *timepoint*	Sets the timepoint in the file where playback begins. Enter the *timepoint* in HH:MM:SS format. Default is 00:00:00

5.9. Summary of TLogger Run-Time Commands

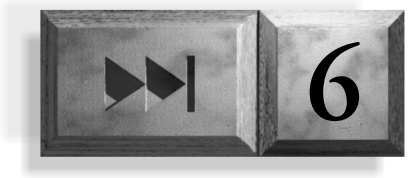
Table 5-2 lists the commands you can send to TLogger while it is running.

Table 5-2: TLogger Run-Time Commands (Sheet 1 of 2)

Command	Description
	Record Mode and Play Mode
`go`	Starts or resumes recording or playback from a previously stopped or paused state.
`stop`	Halts playback at the current timepoint, stops recording or transmission.
`quit`	Closes the TLogger file and exits.
`info` or `i`	Displays a brief status report including the log file name, current timepoint, timescale, and current play state.
`h, help,` or `?`	Displays a summary of TLogger run-time commands.
`wait` *`time`*	Used only in scripts. Specifies how long the TLogger performs its current action before processing the next command in the script. Enter `time` in the HH:MM:SS format or as a number of seconds.
	Record Mode Only
`start`	Sets the current timepoint to the beginning of the recording.
`end`	Sets the current timepoint to the end of the recording.
`pause`	Pauses playback at the current timepoint. In DIS, begins repeated resending of each entity's most recent entity state PDU with velocity and acceleration set to zero.
`scale [`*`factor`*`]`	Sets the timescale `factor` that determines playback speed and direction. Use 1.0 (default) for normal speed, 2.0 for double speed, 0.5 for half speed, and negative values for backward playback.
`time timepoint`	Navigates to a specific `timepoint` in the recording. Specify `timepoint` in HH:MM:SS format.
`skip` *`offset`*	Moves the current timepoint forward or backward by the specified time *`offset`*. Specify the offset in HH:MM:SS format or as a number of seconds. A positive offset causes a jump forward in time; a negative offset causes a backward jump.
`timemaint [`*`n`*`]`	Determines whether the TLogger generates up-to-date entity state messages after you jump to a new timepoint via the `time` command. Use a setting of 0 (default) to disable, 1 to enable, or no argument to display the current setting.

Table 5-2: TLogger Run-Time Commands (Sheet 2 of 2)

Command	Description
`skipmaint [`*`n`*`]`	Determines whether the TLogger generates up-to-date entity state messages after you jump to a new timepoint via the `skip` command. Use a setting of 0 to disable, 1 (default) to enable, or no argument to display the current setting.
`heartbeat [`*`sec`*`]`	DIS only. During slow-speed playback, specifies the number of seconds the TLogger allows before generating an interim PDU for a given entity. Default is 5. With no argument, `heartbeat` displays the current setting.
`timeout [`*`sec`*`]`	DIS only. During slow-speed playback, specifies how long the TLogger continues to transmit interim PDUs for an entity after the entity's PDUs have stopped occurring in the recording. Defaults to 12 recorded seconds.



Controlling the Logger Remotely

You can control the Logger from a remote application via DIS or HLA. Every command available on the Logger graphic interface is available for remote control. This chapter discusses the following:

6.1. Remote Control PDUs

Logger Control PDU classes work similarly to the way View Control PDUs work to control the MÄK Stealth. There are 17 Logger Control PDUs - one for each feature that can be activated remotely. All are derived from *DtLgrControlPdu*, which is defined in *lcPdu.h*, and all are defined in header files that begin with "lc". You can view the header files in the on-line version of the *VR-Link Developer's Guide* for VR-Link version 3.2.

The remote control classes are:

- *DtLgrEndPdu*
- *DtLgrEofActionPdu*
- *DtLgrHeartbeatPdu*
- *DtLgrPausePdu*
- *DtLgrPlayPdu*
- *DtLgrPlayActionPdu*
- *DtLgrPbFilePdu*
- *DtLgrQuitPdu*
- *DtLgrRecordActionPdu*
- *DtLgrRecFilePdu*
- *DtLgrRecordPdu*
- *DtLgrSetTimePdu*
- *DtLgrSkipTimePdu*
- *DtLgrStartPdu*
- *DtLgrStopPdu*
- *DtLgrTimeScalePdu*
- *DtLgrTimeoutPdu*

HLA only

For HLA, *DtLgrControlInteraction*, defined in *hLcInter.h*, is a wrapper around *DtLgrControlPdu*.

6.2. Logger Commands for Remote Control

The remote control command-line arguments control the following behavior:

- Run Logger without the graphic interface

  Since the Logger can be operated remotely, the graphic interface is not a requirement, so you can run the Logger without the interface using the `-v` argument.

- Specify which remote control PDUs the Logger will accept

  You can specify whether or not the Logger will respond to all PDUs, specific PDUs, or no PDUs, with the `-L` argument.

- Record remote control PDUs

  You can keep a record of the control PDUs sent to the Logger. Use the `-R` argument.

DIS only

- Specify an entity ID for Logger

  The Logger will be listening for instructions. It must be addressable so an entity ID is used. You specify the ID with the `-a` argument.

Table 6-1 describes each remote control argument to the Logger command and its possible values.

Table 6-1: Logger command arguments for remote control

Logger command arguments	MTL Syntax	Description
`-a "0:0:0"`	`entityIdStr`	Specify the site:host:entity for Logger
`-v {0|1}`	`LgrApp::visible`	1 = Logger interface visible (default) 0 = Logger interface not visible
`-R {0|1}`	`LgrAIF::recordLc`	0 = do not record Logger Control PDUs (default) 1 = record Logger Control PDUs
`-L {0|1|2}`	`LgrAIF::processLc`	0 = Do note respond to Logger Control PDUs 1 = interpret Logger Control PDUs only if they are addressed to the entity ID specified by the -a argument (default) 2 = interpret all Logger Control PDUs

6.3. Example of Remote Control for a DIS Exercise

The following example demonstrates programmatic remote control of the Logger:

```
/*************************************************************
** Copyright (c) 1997 MaK Technologies, Inc.
** All rights reserved.
*************************************************************/
#include "stdio.h"
#include "exerciseConn.h"
#include "ProcControl.h"
#include "EntityTypes.h"

// Include each LoggerControl header
#include "lcEnd.h"
#include "lcEofAction.h"
#include "lcHeartbeat.h"
#include "lcInter.h"
#include "lcPause.h"
#include "lcPdu.h"
#include "lcPlay.h"
#include "lcPlayAction.h"
#include "lcPlayFile.h"
#include "lcQuit.h"
#include "lcRecAction.h"
#include "lcRecFile.h"
#include "lcRecord.h"
#include "lcSetTime.h"
#include "lcSkipTime.h"
#include "lcStart.h"
#include "lcStop.h"
#include "lcTimeScale.h"
#include "lcTimeout.h"


char * helpstr =
"    PDUTYPE       OPT_ARG\n"
"0   PbFile        filename\n"
"1   RecFile       filename\n"
"2   Quit          \n"
"3   Rewind        \n"
"4   End           \n"
"5   Play          \n"
"6   Pause         \n"
"7   Stop          \n"
"8   Record        \n"
"9   SetTime       float\n"
"10  SkipTime      float\n"
"11  TimeScale     float\n"
"12  PlayAction    {PLAY=0, PAUSE=1, STOP=3}\n"
"13  RecordAction  {RECORD=2, STOP=3}\n"
"14  EofAction     {PAUSE=1, STOP=3, LOOP=5}\n"
"15  HeartBeat     float\n"
"16  Timeout       float\n"
"-1  Quit this application\n"
```

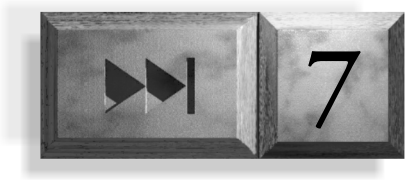
```
;

//
// Application
//
main()
{
   // Create a connection to the exercise or federation execution
   int port          = 3000;
   int exerciseId    = 1;
   int siteId        = 39;
   int applicationNum = 16;

   // Identify the receiving (remote) Logger
   DtObjectId target("9:8:7");

   // Optionally identify yourself
   DtObjectId myId("3:4:5");

   // Create Exercise Connection
   DtExerciseConn exConn(port, exerciseId, siteId, applicationNum);

   // Initialize simulation time
   DtTimeInit();

   // Init work variables
   char  cmdline[128], optionalArg[128];
   int cmd = 0;

   while (cmd >= 0)
   {
      // Print usage instructions and prompt for input
      printf("%s\n> ",helpstr);
      gets (cmdline);

      // Scan for input
      cmd = 0 ; optionalArg[0] = NULL;
      sscanf(cmdline, "%d%s",&cmd,optionalArg);

      // Construct and send desired PDU
      switch(cmd)
      {
      case DtLgrPbFile:
         // Set remote Logger playback-file
         exConn.sendStamped(DtLgrPbFilePdu(myId, target,
                 optionalArg));
         break;

      case DtLgrRecFile:
         // Set remote Logger record-file
         exConn.sendStamped(DtLgrRecFilePdu(myId, target,
                 optionalArg));
         break;

      case DtLgrQuit:
```

```
      // Terminate remote Logger
      exConn.sendStamped(DtLgrQuitPdu(myId,target));
      break;

   case DtLgrStart:
      // Rewind remote Logger (during playback)
      exConn.sendStamped(DtLgrStartPdu(myId,target));
      break;

   case DtLgrEnd:
      // Fast-forward remote Logger to end (during playback)
      exConn.sendStamped(DtLgrEndPdu(myId,target));
      break;

   case DtLgrPlay:
      // Start sending recorded traffic onto network
      // (during playback)
      exConn.sendStamped(DtLgrPlayPdu(myId,target));
      break;

   case DtLgrPause:
      // Pause sending recorded traffic onto network
      // (during playback)
      exConn.sendStamped(DtLgrPausePdu(myId,target));
      break;

   case DtLgrStop:
      // Stop sending recorded traffic onto network
      // (during playback)
      exConn.sendStamped(DtLgrStopPdu(myId,target));
      break;

   case DtLgrRecord:
     // Start recorded traffic from network (during record mode)
      exConn.sendStamped(DtLgrRecordPdu(myId,target));
      break;

   case DtLgrSetTime:
      // Jump to specified time during playback
      exConn.sendStamped(DtLgrSetTimePdu(myId,target,atof
              (optionalArg)));
      break;

   case DtLgrSkipTime:
      // Move forward or backwards by specified amount of time
           // during playback
      exConn.sendStamped(DtLgrSkipTimePdu(myId,target,
              atof(optionalArg)));
      break;

   case DtLgrTimeScale:
      // Control the speed of playback
      exConn.sendStamped(DtLgrTimeScalePdu
              (myId,target,atof(optionalArg)));
      break;
```

```
            case DtLgrPlayAction:
               // Action to take when entering play mode

              exConn.sendStamped(DtLgrPlayActionPdu(myId,target,
                      (DtLgrMode)atoi(optionalArg)));
               break;

            case DtLgrRecordAction:
               // Action to take when entering record mode

               exConn.sendStamped(DtLgrRecordActionPdu(myId,target,
                       (DtLgrMode)atoi(optionalArg)));
               break;

            case DtLgrEofAction:
               // Action to take when play mode reaches end of tape

               exConn.sendStamped(DtLgrEofActionPdu(myId,target,
                       (DtLgrMode)atoi(optionalArg)));
               break;

            case DtLgrHeartbeat:
               // set heartbeat on remote Logger
               exConn.sendStamped(DtLgrHeartbeatPdu
                       (myId,target,atof(optionalArg)));
               break;

            case DtLgrTimeout:
               // Set DIS timeout value for remote Logger
               exConn.sendStamped(DtLgrTimeoutPdu(myId,target,atof
                       (optionalArg)));
               break;
            }
         }
      }
```

Note: A program can use wildcards to identify the Loggers it sends commands to. For example:

```
DtObjectId target("255:255:7");
```



Logger Tools

The Logger Tools are utilities that let you combine, modify, and inspect recorded Logger tapes. This chapter describes:

Like the Logger itself, the Logger Tools are provided in separate versions for HLA and the supported DIS revisions.

7.1. Viewing Logger Files with lgrDump

The *lgrDump* utility presents the contents of a Logger file in a readable form. The output format is very similar to that of the *netdump* and *hlaNetdump* utilities included with the VR-Link Toolkit.

Usage:

```
lgrDump [filename]
```

Because *lgrDump* directs its output to standard output, you can redirect the output to a destination file or program as in these examples:

```
lgrDump logfile.lgr > output.txt
lgrDump logfile.lgr | more
```

> To pipe output from another program to *lgrDump*, omit the *filename* argument as in these examples:

```
lgrMerge File1.lgr File2.lgr | lgrDump
cat exercise_1.lgr | lgrDump
```

Note: Owners of the VR-Link Toolkit can filter messages by specified field values by piping *lgrDump's* output through the Toolkit's *buffgrep* utility. See the VR-Link documentation for more information.

The following text shows a sample PDU as displayed by the DIS version of *lgrDump*.

```
*********************************************************
******** PDU #1077   Size=144         39.016 secs *******
*********************************************************
PduKind:             EntityStatePduKind   (1)
Version:             4
Exercise:            1
ProtocolFamily:      FamilyEntityInteraction   (1)
TimeStamp:           3297.213(seconds past hour)
TimeStampType:       Relative
Size:                144(bytes)
From:                1:15:1  (site:application:entity)
ForceID:             1
NumArtParams:        0
NumAttachedParts:    0
Type:    1:2:225:2:1:0:0 (kind:domain:country:cat:subcat:specific:extra)
Guise:   1:2:225:2:1:0:0 (kind:domain:country:cat:subcat:specific:extra)
Location:            { -2661471.288,  -4358184.189,   3809002.286}
Velocity:            {       37.102,        87.402,       127.072}
Orientation:         {        1.169,        -0.929,        -2.305}
Acceleration:        {        0.000,         0.000,         0.000}
AngularVel:          {        0.000,         0.000,         0.000}
DrAlgorithm:         DrDrmRvw   (4)
Appearance:          0x00000000
   PaintScheme       _PaintSchemeUniform
   MobilityKill      _MobilityKillNone
   FirePowerKill     _FirePowerKillNone
   Damage            _DamageNone
   Smoke             _SmokeNone
```

```
    TrailingEffects   _TrailingEffectsNone
    Hatch             _HatchNA
    Lights            _LightsNone
    Flaming           _FlamingNo
    FinalPdu          _NotFinalPdu
    AfterBurner       UnknownEnum
 Marking:             A10
 CharSet:             1
 Capabilities:        0x00000000
```

Each PDU description begins with a heading containing a PDU number, packet size, and time elapsed since the start of the Logger tape. This information is added by the Logger during recording, and is not a part of the actual PDU.

After the heading, *lgrDump* displays all fields of the PDU.

lgrdump prints version information as described in Table 7-1:

Table 7-1: PDU versions and DIS protocol versions

PDU Version	DIS Protocol Version
3	2.0.3
4	2.0.4
5	IEEE 1278.1
6	2.1.4

7.1.1. *lgrDump* for HLA

If you are using the DMSO RTI, *lgrDump* requires *rtiexec*. See "Obtaining and Installing a Runtime Infrastructure (RTI)", on page 2-10 for more information about *rtiexec*.

The class, attribute and parameter handles printed by the HLA version of *lgrDump* may not be the actual handles that were originally recorded if you are not using the same *.fed* file with *lgrDump* that you used when you recorded the file with the Logger. The handle numbers that are printed are those associated with the recorded names in the new FOM.

If you are using a different *.fed* file with *lgrDump*, but would like to see the actual handles that were used at the time of recording, you can use *lgrDump's* `-x` argument with an *execName* of "orig". This is only allowed when using *lgrDump* in raw mode. For example:

```
lgrDump -r -x orig myFile.lgr
```

In non-raw mode, we need to actually interpret the data, so we need to use the handles associated with the current *.fed* file.

7.1.2. How *lgrDump* Displays Unexpected Message Types

Like the Logger itself, this package provides separate *lgrDump* versions for HLA, DIS 2.0.3 and 2.0.4/IEEE-1278.1/2.1.4. Each *lgrDump* executable displays formatted data for messages of its intended protocol.

DIS only

If the DIS version of *lgrDump* encounters a PDU of an incorrect DIS version (or a non-DIS PDU), it displays available header information followed by a hex dump of the rest of the PDU.

7.1.3. Displaying Raw Data

When you run *lgrDump* with the *-r* option (for raw), it behaves like the *nethex* utility that ships with the VR-Link toolkit. For each attribute update or interaction received, it prints the name of the attribute, the size of the value of the attribute, and a hexadecimal dump of the value of the attribute.

7.2. Concatenating Files with lgrCat

The *lgrCat* utility concatenates together messages from a series of two or more Logger files. The resulting file contains all messages from the first file followed by all messages from the second file, and so on.

In the output file, messages from the first file retain their original times and message numbers. For each subsequent file, *lgrCat* increases the starting time and message numbers to create a continuous message sequence, as in Figure 7-1.

```
If File1.lgr contains:            ...and File2.lgr contains:
         Msg #1 at time  5.0              Msg #1 at time 2.0
         Msg #2 at time 10.0              Msg #2 at time 4.0
         Msg #3 at time 20.0              Msg #3 at time 6.0

                    ...then the concatenated file contains:
                        Msg #1 at time  5.0
                        Msg #2 at time 10.0
                        Msg #3 at time 20.0
                        Msg #4 at time 22.0
                        Msg #5 at time 24.0
                        Msg #6 at time 26.0
```

Figure 7-1. *lgrCat* example

lgrCat modifies only the message numbers and time values generated by the Logger (those that appear in the message heading when viewed with *lgrDump*).

DIS only

Under DIS, *lgrCat* does not change the TimeStamp values contained in original PDUs.

HLA only

When you use the HLA version of *lgrCat*, all of the input Logger files must have been recorded using the same FOM. Files recorded using FOMs that differ from that used in the first file on the command line are skipped.

Usage:

```
lgrCat [-s spacing] file1 [file2 file3...]
```

lgrCat directs its output to standard output (usually the display) in a fashion similar to the UNIX *cat* or DOS *type* command. In most cases, you redirect or pipe the output to a destination file or program, and specify two or more files to be concatenated, as in these examples:

```
lgrCat file1 file2 [file3 file4...] > output.file
lgrCat exercise1.lgr exercise2.lgr > ex_1-2.lgr
lgrCat ex1.lgr ex2.lgr ex3.lgr | lgrDump | more
```

7.2.1. Inserting Time Between Concatenated Files

You can use the optional `-s spacing` argument to insert a period of time between files. The `spacing` argument specifies the number of seconds to insert between the final message in each file and the starting point of the next file (timepoint 00:00:00). For example, the command,

```
lgrCat -s 4.0 File1.lgr File2.lgr > combined.lgr
```

creates a *combined.lgr* file whose first three messages match those in the output file in Figure 7-1. Now, however, the last three messages have been delayed by four seconds and now occur at 26.0 28.0 and 30.0 seconds, as in Figure 7-2:

```
If File1.lgr contains:            ...and File2.lgr contains:
         Msg #1 at time  5.0              Msg #1 at time 2.0
         Msg #2 at time 10.0              Msg #2 at time 4.0
         Msg #3 at time 20.0              Msg #3 at time 6.0

                    ...then the concatenated file will contain:
                         Msg #1 at time  5.0
                         Msg #2 at time 10.0
                         Msg #3 at time 20.0
                          (four-second space)
                         Msg #4 at time 26.0
                         Msg #5 at time 28.0
                         Msg #6 at time 30.0
```

Figure 7-2. Effect of the *lgrCat* spacer argument

7.2.2. Processing Input from Other Programs

> To pipe the output of another program to *lgrCat*, replace one of *lgrCat*'s input file arguments with a hyphen character to specify where the incoming data should appear in the concatenated output.

In the following example, *lgrCat* receives output piped from the *lgrOffset* program as its first input (indicated by the hyphen), and concatenates that data with *File2.lgr*. The combined data is directed to *Out.lgr*:

```
lgrOffset 20.0 < File1.lgr | lgrCat - File2.lgr > Out.lgr
```

7.3. Merging Files with lgrMerge

The *lgrMerge* utility combines one or more Logger files into a single file containing all messages from all files, sorted in chronological order and renumbered. Figure 7-3 illustrates a merge of two Logger files:

```
If File1.lgr contains:            ...and File2.lgr contains:
         Msg #1 at time 1.0                Msg #1 at time 2.0
         Msg #2 at time 3.0                Msg #2 at time 4.0
         Msg #3 at time 5.0                Msg #3 at time 6.0

                        ...then the merged file contains:
                          Msg #1 at time 1.0
                          Msg #2 at time 2.0
                          Msg #3 at time 3.0
                          Msg #4 at time 4.0
                          Msg #5 at time 5.0
                          Msg #6 at time 6.0
```

Figure 7-3. How *lgrMerge* combines files

lgrMerge modifies only the message numbers and time values generated by the Logger (those that appear in the message heading when viewed with *lgrDump*).

DIS only

Under DIS, *lgrMerge* does not change the TimeStamp values contained in the original PDUs.

HLA only

When you use the HLA version of *lgrMerge*, all of the input Logger files must have been recorded using the same FOM. Files recorded using FOMs that differ from that used in the first file on the command line are skipped.

Usage:

```
lgrMerge file1 file2 [file3 file4...]
```

Because *lgrMerge* directs its output to standard output, you typically redirect or pipe its output to a destination file or program as in these examples:

```
lgrMerge file1 file2 file3 > outputfile
lgrMerge exercise1.lgr exercise2.lgr > ex_1-2.lgr
lgrMerge ex1.lgr ex2.lgr ex3.lgr | lgrDump > PDUs.txt
```

7.4. Using lgrOffset to Shift Messages in Time

The *lgrOffset* utility can adjust the time of all messages in a file by a specified number of seconds. You can:

- Decrease or increase the timepoints of all messages in a file so they occur earlier or later.
- Remove an unknown length of dead time from the beginning of a file.

During recording, the Logger records the arrival time of each message relative to the start of recording. This is the only value that *lgrOffset* modifies; it is not a part of the original message. The original TimeStamp values in the messages are not affected.

Usage:

```
lgrOffset time|first
```

where *time* is the desired time offset in seconds. *time* can be a positive or negative value.

> To remove a period of dead time from the beginning of a file, use the `first` keyword as the argument instead of a numeric `time` value. This causes *lgrOffset* to determine the timepoint of the first message, and to subtract that number of seconds from all messages in the file.

Because *lgrOffset* receives its input from standard input and directs its output to standard output, piping or redirection are used to specify the input source and output destination.

In either of the following two examples, *lgrOffset* increases the timepoints of all messages in *infile.lgr* by 15 seconds and saves the output in *outfile.lgr*:

```
lgrOffset 15.0 < infile.lgr > outfile.lgr
lgrCat infile.lgr | lgrOffset 15.0 > outfile.lgr
```

In the next example, *lgrOffset* removes an unknown amount of dead time from the beginning of *infile.lgr* and saves the output in *outfile.lgr*:

```
lgrOffset first < infile.lgr > outfile.lgr
```

7.5. Using lgrTrim to Extract Part of a File

The *lgrTrim* utility reads and outputs a specified portion of a Logger file, in effect, trimming packets from the beginning and end of the file. You can specify the part of the file to be output, as either a range of messages or a span of time.

Usage:

```
lgrTrim [-n] start [end]
```

The `start` and `end` arguments determine the part of the file to be included in the output. *lgrTrim* discards any packets that fall outside of the specified range. If no `end` is specified, *lgrTrim* outputs all packets from the specified `start` point to the end of the file.

By default, *lgrTrim* assumes that the `start` and `end` arguments are time values, given in seconds.

> To specify a numerical range of messages as the `start` and `end` points, use the `-n` option.

Because *lgrTrim* receives its input from standard input and directs its output to standard output, piping or redirection are used to specify the input source and output destination.

Example 1

lgrTrim extracts from *infile.lgr* all messages that occur between times 00:00:05 and 00:00:10 and saves them in *outfile.lgr*:

```
lgrTrim 5.0 10.0 < infile.lgr > outfile.lgr
```

Example 2:

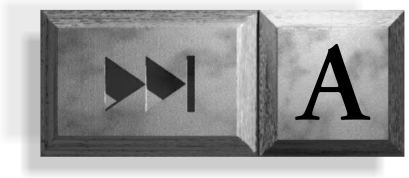
lgrTrim extracts the 10th through 20th messages from *infile.lgr* and saves them in *outfile.lgr*:

```
lgrTrim -n 10 20 < infile.lgr > outfile.lgr
```

Example 3:

lgrTrim output piped to *lgrDump* for on-screen display of the 30th through 35th messages in *exerc01.lgr*:

```
lgrTrim -n 30 35 < exerc01.lgr | lgrDump
```



Troubleshooting

This chapter describes how to identify and correct the most common network problems that prevent the Logger from sending and receiving data properly:

- Ensuring the DIS Logger uses the correct IP address and UDP port (Troubleshooting Logger Network Communications, on page A-1)
- Specifying the correct network mask in Windows (Configuring the Network Mask (Windows only), on page A-3)

A.1. Troubleshooting Logger Network Communications

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 command line or configuration file at startup. For more information, see your System Administrator, or use the UNIX `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 -A` *`address`* `-P` *`portnum`*

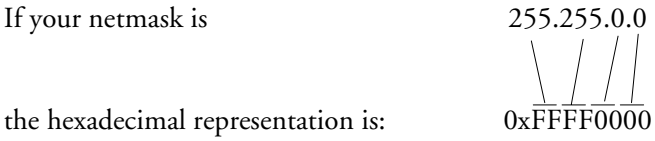
   where address and portnum are the broadcast address and UDP port used by your other applications, for example,

   ```
   Logger -A 123.456.999.255 -P 3020
   ```

A.1.2. Configuring the Network Mask (Windows only)

The Windows versions of the Logger and TLogger expect to run on a class C network using a netmask of 255.255.255.0. If your machine is configured to use a different netmask, create a DtNetmask environment variable to reflect your netmask.

The DtNetmask variable should specify your netmask in a C-language style hexadecimal format. For example:



The default value of DtNetmask is 0xFFFFFF00, which corresponds to the netmask 255.255.255.0.

To set the DtNetmask variable for Windows NT:

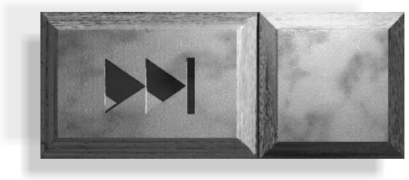
1. Open the Windows NT Control Panel.
2. Open the System applet.
3. Enter the DtNetmask variable name and appropriate hexadecimal value in the fields at the bottom of the System dialog box.
4. Click Set.

To set the DtNetmask variable for Windows 95:

1. Add the following line to your AUTOEXEC.BAT file. Replace the hexadecimal number with one that represents your netmask:

   ```
   SET DtNetmask=0xFFFF0000
   ```

2. Reboot.



Glossary

For additional terms and definitions, see the HLA glossary at:
http://hla.dmso.mil/hla/general/hlagloss.html

dead-reckoning

A method of approximating the location of an object based on its direction and velocity. The Logger uses dead-reckoning to generate interim state messages for entity maintenance.

Defense Modeling and Simulation Office

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

DIS

Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

Distributed Interactive Simulation

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

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

DMSO

Defense Modeling and Simulation Office.

entity

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

entity maintenance

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

exercise

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

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

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

federate

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

federation

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

Federation Object Model

Defines the data content of a federation execution.

federation execution

The federation execution represents the actual operation, over time, of a subset of the federates and the RTI initialization data taken from a particular federation. It is the step where the executable code is run to conduct the exercise and produce the data for the measures of effectiveness for the federation execution. A federation execution is the logical equivalent of a DIS simulation exercise.

field of view

Field of view controls the perspective of the eyepoint.

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

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

filter range

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

FOM

Federation Object Model

frame rate

The rate at which the Logger displays updated images.

heartbeat

The frequency with which the Logger sends interim PDU messages. (DIS only)

High-Level Architecture

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

HLA

High-Level Architecture.

interaction

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

Logger control PDUs

A set of classes that you can use to control the Logger programmatically. Available in both DIS and HLA.

MÄK Technologies Lisp

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

mode

Refers to the Logger's current activity or readiness state, either recording, or playing. The Logger can do only one of these activities at a time.

MTL

MÄK Technologies Lisp.

object

An element in a simulation. Also called an entity.

PDU

Protocol Data Unit

Protocol Data Unit

A unit of data that is passed on a network between DIS simulation applications.

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

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

reference FOM

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

RTI Initialization Data (RID)

The data required by the RTI for operation. The required data come from two distinct sources, the Federation Object Model product, and the Federation Execution Details.

RPR FOM

Realtime Platform Reference FOM

RTI

Run-Time Infrastructure

Run-Time Infrastructure

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

state

Within each mode, refers to whether the Logger is stopped, paused, or active (recording or playing.)

tape

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

timeout

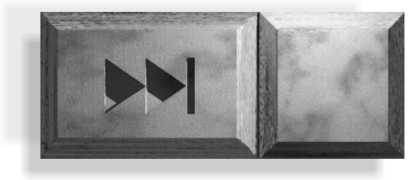
The period of time in which the Logger continues to display an entity after that entity's update messages have stopped appearing on the network.

timescale

A factor by which time is accelerated or slowed in

UDP port

A network channel through which the Logger sends and receives data for DIS exercises.



Index

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X-Y-Z

MÄK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138
617.876.8085

LOG-3.3-1-981007