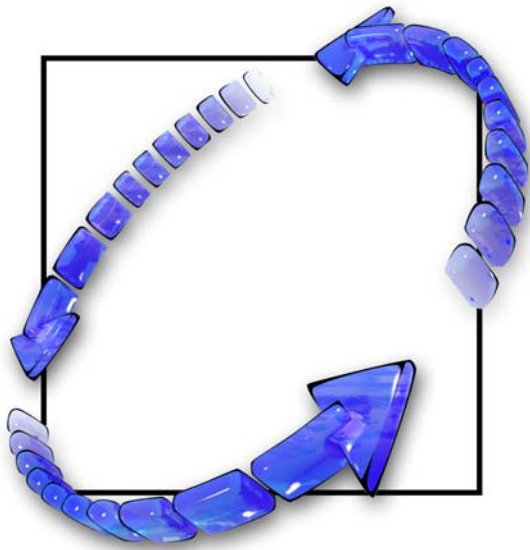




VR-Link



Getting Started Guide



VR-Link



Getting Started Guide

Copyright © 2012 VT MÄK
All rights Reserved. Printed in the United States.

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

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

All other trademarks are owned by their respective companies.

For third-party license information, please see “Preface,” on page v.

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

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

info@mak.com
www.mak.com

Revision VRL-4.0.6-4-121031



Contents

Preface

How this Manual is Organized	v
VR-Link Documentation Set	v
MÄK Products	vi
How to Contact Us	viii
Document Conventions	ix
Mouse Button Naming Conventions.....	x
Third Party Licenses	x
Boost License.....	x
libXML and libICONV	xi

Chapter 1 Introduction

1.1. The Protocol-Independent API	1-2
1.2. VR-Link Features	1-2
1.2.1. Files and Executables in VR-Link	1-4
1.3. Simulation Standards Supported	1-4
1.3.1. Support for the HLA RTI 1.3 and RTI 1516 Specifications ...	1-5
1.3.2. HLA FOM Support	1-6
1.4. Developer Documentation	1-6

Chapter 2 Installing and Configuring VR-Link

2.1. Installing VR-Link	2-2
2.1.1. Installing VR-Link on Windows	2-2
2.1.2. Installing VR-Link on a Linux System	2-2
2.2. VR-Link Files Installed	2-3
2.3. Installing and Setting Up the MÄK License Manager	2-4
2.3.1. Specifying the License Server	2-5
2.4. Installing an RTI	2-7
2.4.1. Installing the MÄK RTI	2-8

Contents

2.5. Configuring the Network for DIS Applications 2-9

2.5.1. Broadcast Address and Netmask (UNIX) 2-10

2.5.2. UDP Port 2-10

MÄK Products Glossary

Index



Preface

This guide is a brief introduction to VR-Link to get you started. For developer documentation, please see the online API documentation.

We are constantly improving VR-Link. For the most recent information, please read the release notes distributed with your version of VR-Link.

How this Manual is Organized

The chapters are organized as follows:

Chapter 1, *Introduction*, describes the main features of VR-Link and contains information about simulation standards.

Chapter 2, *Installing and Configuring VR-Link*, explains how to install VR-Link and other required and optional software.

The *MÄK Products Glossary* defines important terms.

VR-Link Documentation Set

The VR-Link documentation set is as follows:

- ♦ *VR-Link Getting Started Guide*. A brief introduction to VR-Link and installation instructions.
- ♦ API documentation. Developer documentation and class documentation. Classes are documented by protocol in HTML files built from the header files. The starting point for the class reference is *vrlink4.0.6/doc/reference/index.html*.
- ♦ *VR-Link Release Notes*. Release information and last minute updates.

The *VR-Link Getting Started Guide* and *VR-Link Release Notes* are provided as PDF files in the *.doc* directory. To view and print PDF documents, you need a copy of the free Acrobat Reader software (version 5.0 or later). It is available at www.adobe.com.

MÄK Products

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

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

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

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

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

- ♦ VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes four end-user applications (VR-Vantage Stealth, VR-Vantage XR, VR-Vantage PVD, and VR-Vantage IG), the VR-Vantage Toolkit, and VR-Vantage FreeView.
 - VR-Vantage Stealth displays a realistic, 3D 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.
 - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
 - VR-Vantage XR provides a common operational picture using the same 3D views as VR-Vantage Stealth, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world.
 - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
 - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
 - VR-Vantage Free View is a terrain and 3D model viewer that introduces some of the features of VR-Vantage applications in a useful, free utility.
- ♦ MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- ♦ VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- ♦ VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.

- ♦ VR-inTerra. VR-inTerra is a C++ API for adding terrain agility to applications. It can load, page, or stream terrain from a wide variety of formats or sources into a single, consistent run-time representation, consisting of a collision graph and vector network.

How to Contact Us

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

Telephone

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

E-mail

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

Internet

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

Post

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

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

Document Conventions

This manual uses the following typographic conventions:

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

Directory names are preceded with dot and slash characters that show their position with respect to the VR-Link home directory. For example, the directory *vrlink4.0.6/doc* appears in the text as *./doc*.

Mouse Button Naming Conventions

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

Third Party Licenses

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

Boost License

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

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

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

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

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

libXML and libICONV

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

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



Introduction

The VR-Link toolkit is an object-oriented library of C++ classes, functions, and definitions that minimize the effort required to create networked simulators and virtual reality applications. VR-Link's powerful, easy-to-use programmer's interface greatly reduces development cost, time, and risk.

The Protocol-Independent API	1-2
VR-Link Features	1-2
Files and Executables in VR-Link	1-4
Simulation Standards Supported	1-4
Support for the HLA RTI 1.3 and RTI 1516 Specifications	1-5
HLA FOM Support	1-6
Developer Documentation	1-6

1.1. The Protocol-Independent API

With VR-Link's protocol-independent API, you can simulate local entities, set their state, and automatically send entity information to other applications over a network using Distributed Interactive Simulation (DIS) or the High-Level Architecture (HLA) Run-Time Infrastructure (RTI). VR-Link also simplifies receiving and processing information from other applications, providing easy access to the state of remote entities. VR-Link handles dead reckoning, thresholding, responding to attribute requests, filtering, and many other tasks.

Because VR-Link supports both DIS and HLA through very similar APIs, you can often switch your applications between the two by changing just a few lines of initialization code, and recompiling. This means that your VR-Link-based applications can maintain the DIS compliance vital to ongoing projects, while migrating to HLA.

DIS only

VR-Link implements all DIS PDUs and you can add support for user-defined PDUs.

1.2. VR-Link Features

VR-Link provides a range of features to help you create and maintain HLA and DIS applications:

- **Exercise Connection:** An exercise connection is a VR-Link application's interface to an HLA or DIS exercise. It provides a protocol-independent¹ interface through which to exchange simulation information with other applications, either through the DIS network, or the HLA RTI.
- **Object tracking:** A VR-Link application uses a reflected entity list to keep track of remote participants in your virtual world by processing incoming attribute updates (through either DIS or HLA), and provides a protocol-independent¹ interface to their state. It submits update requests as needed, provides notice when an entity enters or leaves an exercise, and performs dead-reckoning, trajectory smoothing, and filtering, as desired. Reflected object lists are available for other types of objects, such as emitters, transmitters, and so on.
- **Object publishing:** A VR-Link application uses an Entity Publisher to keep remote applications informed about the state of entities that you simulate locally. You periodically set the current state of your objects through its protocol-independent interface, and the Entity Publisher automatically sends state updates when data changes or exceeds configurable thresholds. Object publishers are available for other types of objects, such as emitters, transmitters, and so on.

1. The interface is mostly protocol-independent. You must compile for the simulation standard on which you will run the application. Some classes have protocol-specific constructors.

- ♦ Interaction classes: VR-Link provides a protocol-independent¹ interface to the sending and receiving of interaction messages that describe simulation events such as weapon fires, detonations, and radio signal transmissions. C++ classes representing DIS PDUs and HLA interactions usually provide mutator and inspector functions to access each field. Variable length PDUs and parameters are resized transparently. Functions are provided to print human-readable representations of interaction data.
- ♦ FOM-Agility: While VR-Link comes with built-in support for the RPR FOM, the FOM Mapper class lets you map VR-Link's existing protocol-independent API to another FOM's parameters, attributes, and object or interaction classes. In this way, code that uses the API does not need to change when the FOM changes.
- ♦ User extensibility: VR-Link's C++ API and implementation allow you to override most of its default functionality through subclassing. You can extend the toolkit to work with new types of HLA object or interaction classes and with new user-defined DIS PDUs. The VR-Link Code Generator can automatically generate the files needed to create new objects and interactions for HLA based on an XML or OMT data definition file.
- ♦ Access to low-level details: For developers who want to work below the level of abstraction provided by our top-level API, VR-Link provides protocol-specific classes and functions. Low-level access includes direct access to the HLA RTI and to the details of network configuration in DIS.
- ♦ Utility functions: VR-Link includes a rich set of utility functions, including vector and matrix manipulation functions, a platform-independent interface to the system clock, and support for discrete simulation time. Coordinate conversion utilities for geocentric coordinates, geodetic coordinates, topographic coordinates, and UTM coordinates are included as well.
- ♦ Example applications: VR-Link comes with a set of HLA and DIS utility programs. For example, the *netdump* utility prints data received from remote simulations in an easy-to-read format. The *f18* utility is a simple networked simulator that serves as a flexible debugging tool. Source code for the examples is provided to demonstrate the use of much of VR-Link's functionality.
- ♦ You can use VR-Link's C++ interface from C applications.

1.2.1. Files and Executables in VR-Link

VR-Link includes the header files and libraries necessary to build applications based on VR-Link, plus executables and source code for the following utility applications:

- ♦ *f18*
- ♦ *netdump*
- ♦ *talk*
- ♦ *listen*.

VR-Link includes source code and executables for the sample applications. The “test” applications show how to extend the features of VR-Link by creating your own PDUs, interactions, encoders, and decoders.

1.3. Simulation Standards Supported

VR-Link supports HLA and DIS. For a list of the versions of the simulation standards supported by your release of VR-Link, please refer to your release documentation.

You can also check the MÄK web site for platform support updates at:
<http://www.mak.com/support/product-versions.html>.

For a brief discussion of DIS and HLA, please see Section 2.2, “*HLA, DIS, and Protocol Independence*,” in the API documentation.

1.3.1. Support for the HLA RTI 1.3 and RTI 1516 Specifications

VR-Link supports the HLA 1.3 specification, the HLA 1516 specification (SISO DLC HLA API), and the HLA Evolved (IEEE 1516-2010) specification. VR-Link's protocol independent interface lets you create applications for use with any version of the HLA largely without regard to the details of the RTI. Any code that you have written for use with the RTI 1.3 specification should be usable with RTI 1516 or HLA Evolved with very few, if any, changes.

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

HLA Evolved is the latest version of HLA. It builds on the work done by the SISO DLC HLA API group to provide a dynamic link compatible API. It includes additional features such as FOM modules and update rate reduction.



In most cases, VR-Link's implementation of HLA Evolved is identical to HLA 1516. Therefore, any references to HLA 1516 in this manual should be considered to apply to HLA Evolved as well, unless noted otherwise.

Compatibility of RTI 1.3 and RTI 1516 Applications

If you use the MÄK RTI, applications built using the VR-Link 1516 APIs can interoperate with applications built with the VR-Link 1.3 API. For example, if you build the listen example with the 1516 API and the talk example with the 1.3 API, they can interoperate. For more information, please see Section 5.10, [“Interoperability Between HLA 1.3 and IEEE 1516 Federates,”](#) in the API documentation.

Support for FED and FDD (XML) Files

VR-Link supports HLA configuration files in both the FED and FDD (XML) formats. When you create an exercise connection, you can specify a *.fed* file, *.fdd* file, or *.xml* file. You can use any of these formats with RTI 1516, HLA Evolved, and RTI 1.3.

If you do not pass a filename to the exercise connection, VR-Link uses the federation execution name as the filename. If you are building for RTI 1516, VR-Link looks for a file name *federation_execution.xml*. If it cannot find a file with this name, it looks for a file named *federation_execution.fdd*, then for *federation_execution.fed*. If you are building for RTI 1.3, VR-Link looks for a *.fed* file first, then an *.xml* or *.fdd* file.

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

1.3.2. HLA FOM Support

VR-Link provides extensive built-in support for the Real- Time Platform Reference FOM (RPR FOM). (For version information, please see release documentation.) The RPR FOM is a reference FOM developed by the SISO-sanctioned RPR FOM Standards Development Group, which consists of representatives from many companies that used the DIS protocol in the pre-HLA era.

The goal of the RPR FOM is to facilitate a priori interoperability (to about the extent we had in DIS) among HLA simulations that choose to use it. In other words, if you use the RPR FOM (either as is, or with your own extensions), then you know that you will be able to interoperate with anyone else who chooses to use this FOM. Another advantage of using the RPR FOM is that many simulation tools, including those offered by VT MÄK, support this FOM.

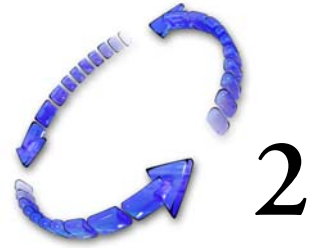
FOM Agility

In addition to its built-in support for the RPR FOM, VR-Link can be configured to work with other FOMs, by using its FOM Mapper to define mappings between VR-Link's protocol-independent interface and the objects, interactions, parameters, and attributes defined in your FOM. For more information about VR-Link's FOM Agility, please see Section 6, [*FOM Agility*](#) in the API documentation.

VR-Link has a FOM Mapper for the MATREX FOM. For details, please contact your MÄK salesperson.

1.4. Developer Documentation

Documentation of the VR-Link APIs is provided as a set of HTML files in [*vrlink4.0.6/doc/reference/index.html*](#). Documentation includes developer documentation that describes the APIs and class documentation that is generated from the comments in the header files.



Installing and Configuring VR-Link

This chapter covers the following topics:

Installing VR-Link.....	2-2
Installing VR-Link on Windows.....	2-2
Installing VR-Link on a Linux System.....	2-2
VR-Link Files Installed	2-3
Installing and Setting Up the MÄK License Manager.....	2-4
Specifying the License Server	2-5
Installing an RTI.....	2-7
Installing the MÄK RTI	2-8
Configuring the Network for DIS Applications	2-9
Broadcast Address and Netmask (UNIX).....	2-10
UDP Port.....	2-10

2.1. Installing VR-Link

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

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

2.1.1. Installing VR-Link on Windows

Note the following:

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

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

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

2.1.2. Installing VR-Link on a Linux System

Before you install VR-Link, please read *VR-Link Release Notes* to see if there are any special instructions for installation.

Linux versions of VR-Link are provided as compressed tar files on DVD, or as downloaded files.

To install VR-Link on Linux:

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

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

where *application* is the product release identification.

2.2. VR-Link Files Installed

The installation process creates a directory called *vrlink4.0.6* and a set of subdirectories beneath it. [Figure 2-1](#) illustrates the directory structure.

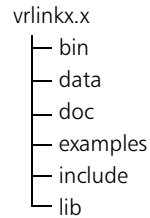


Figure 2-1. VR-Link directory structure

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

Table 2-1: Contents of VR-Link subdirectories

Directory	Directory contents
<i>./bin</i>	Executables, including utilities and example applications.
<i>./data</i>	Miscellaneous data files that are not appropriate for the bin directories.
<i>./doc</i>	VR-Link documentation in PDF format.
<i>./examples</i>	Source directories for the example applications.
<i>./include</i>	C + + header files.
<i>./lib</i>	VR-Link C + + libraries.

On PCs, the default installation directory is *C:/MAK/vrlink4.0.6*.

2.3. Installing and Setting Up the MÄK License Manager

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



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

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

- ♦ Windows: <ftp://ftp.mak.com/out/MAKLicenseManager-win-setup.exe> (32 bit systems)
<ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe> (64 bit systems)
- ♦ Linux: <ftp://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz>

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

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

2.3.1. Specifying the License Server

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

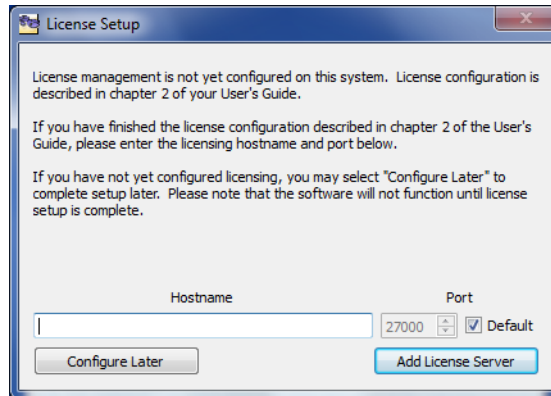


Figure 2-2. License Setup dialog box

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

If you know the hostname, type it in the **Hostname** box. Then click **Add License Server**. The application will start.

i

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

The MAKLMGRD_LICENSE_FILE Environment Variable

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

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

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

Windows

To add the MAKLMGRD_LICENSE_FILE in Windows:

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



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

Linux

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

```
setenv MAKLMGRD_LICENSE_FILE @oak
```

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

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

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

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

2.4. Installing an RTI

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

i

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

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

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

2.4.1. Installing the MÄK RTI

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

Configuring Your System to Use the MÄK RTI

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

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

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

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

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

Running Applications with the MÄK RTI

In many cases, you do not need an RTI server, such as the rtiexec, or a central application to use the MÄK RTI, however you can run the rtiexec if you want to. (It is required to use certain features of the MÄK RTI.) Be sure the license server is running, then start the applications, and they should be able to communicate. For more information, please see your RTI documentation.

2.5. Configuring the Network for DIS Applications

In a typical DIS exercise, VR-Link applications communicate with other DIS applications using broadcast UDP/IP over a local area network. In such a configuration:

- ♦ All applications in the exercise use the same IP broadcast address, and each application's host machine has a network interface device configured for that address.
- ♦ All machines must share a common netmask.
- ♦ All participating applications must use the same UDP port.

Typically, machines on a LAN will already have the correct broadcast addresses, so VR-Link applications will be able to communicate with each other immediately. However, if applications are unable to achieve two-way communication, the network devices may be improperly configured.

2.5.1. Broadcast Address and Netmask (UNIX)

Network devices should be configured so that all machines agree on the netmask and on the part of the network address (inet) covered by the netmask. On UNIX-based systems, you can use the `ifconfig` command to examine and set the configuration of network devices. For example, to set the netmask use:

```
ifconfig device_name netmask 0xFFFF0000
```

Some machines have more than one network device capable of broadcasting. Unless your application is designed to specify a specific device or address (which is possible using VR-Link routines), a VR-Link application will use the broadcast address of the first device listed in the interface table by default.

Please see your system administrator if you need help with network issues.

2.5.2. UDP Port

In addition to the broadcast address, the applications in an exercise must also agree on the UDP port number.

The applications that ship with VR-Link use port 3000 by default. Most MÄK applications let you set the port number with the `-P port` option.



MÄK Products Glossary

This glossary defines terms used in MÄK product documentation.

absolute dead reckoning

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

aggregate

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

AMO

Application Management Object (AMO).

Application Management Object (AMO)

An AMO is a standard TENA object that is supposed to be used by every TENA application. It gives other applications information about the local application. The VR-Link exercise connection, by default, publishes an AMO object, updates it at a specified rate (the rate is part of the AMO's state repository; the default is 30 seconds), and updates the number of objects published and proxies (TENA reflected objects) held.

API

Application Programming Interface.

articulated part

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

attached part

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

body coordinate frame

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

channel

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

clipping

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

clipping planes

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

culling

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

Cartesian coordinates

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

dead-reckoning

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

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.

display engine

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

Distributed Interactive Simulation

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

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

DMSO

Defense Modeling and Simulation Office.

emitter

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

entity

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

entity maintenance

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

entity-type

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

The components of an entity-type are:

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

environmental process

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

Euler angles

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

event

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

execution

The TENA term for one or more interacting simulation applications.

Execution Manager

An Execution Manager governs a TENA execution for applications joining, resigning, or changing subscriptions to that execution.

exercise

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

exercise connection

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

exercise ID

A numeric identification for a DIS simulation exercise.

FED file

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

federate

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

federation

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

Federation Object Model

Defines the data content of a federation execution.

federation execution

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

field of view

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 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 application displays updated images.

ground clamping

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

geocentric coordinates

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

geodetic coordinates

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

gridded data

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

guise

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

head-up display

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

heartbeat

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

High-Level Architecture

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

HLA

High-Level Architecture.

interaction

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

Logger control PDUs

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

logical range

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

Local RTI Component

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

Logical Range Object Model

The data definition file for TENA exercises. It contains the set of object and message definitions that may be used in the exercise.

LRC

Local RTI Component

LROM

Logical Range Object Model.

MÄK Technologies Lisp

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

message

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

MTL

MÄK Technologies Lisp.

Network Naming Service

Acts as a registry for TENA Execution Managers.

NNS

Network Naming Service.

orientation clamping

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

object

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

object handle

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

object name

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

PDU

Protocol Data Unit.

Protocol Data Unit

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

proxy

In TENA, a reflected object (Stateful Distributed Object).

radio transmitter

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

radio receiver

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

range resource

A participant in a TENA execution (similar in concept to the HLA federate).

Realtime Platform Reference FOM

An HLA reference FOM based on the DIS protocol.

recording

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

reflected entity list

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

reference FOM

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

relative dead reckoning

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

remote display engine

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

RTI Initialization Data (RID)

The initialization data required by the RTI for operation.

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

RPR FOM

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.

SDO

Stateful Distributed Object.

servant

A published object (SDO) in TENA.

Simulation Interoperability Standards Organization

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

simulation time

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

SISO

Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

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

state

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

Stateful Distributed Object

An object in a TENA exercise.

tape

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

TENA

Test and Training Enabling Architecture.

TENA Middleware

"The software that links range resources together into a logical range. The "TENA Middleware" is the software component that is in between (that is, in the middle of) range resources during a logical range execution." – from www.tena-sda.org

terrain following

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

Test and Training Enabling Architecture

Test and Training Enabling Architecture (TENA) is an interoperability architecture used to pass data in the form of object updates and messages (like HLA objects and interactions) from one application to another.

timeout

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

timescale

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

topographic coordinates

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

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

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

UDP port

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

Universal Transverse Mercator

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

UTM coordinates

Universal Transverse Mercator coordinates.

view control messages

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

view floor

In MÄK Stealth 6.x and earlier, a minimum height above the terrain in Absolute or Track mode, below which the eyepoint is not allowed to go. The view floor is measured relative to the terrain surface.



Index

A

address, broadcast 2-10
application, example 1-4

C

compatibility, 1.3 and 1516 1-5
configuration file, RTI 2-8
configuring
 broadcast address and netmask for UNIX 2-10
 system for MÄK RTI 2-8
 VR-Link for DIS 2-9

D

default, port 2-10
dialog box, License Setup 2-5
directory, tree, VR-Link 2-3
DIS, network configuration 2-9
documentation, location of 2-3

E

environment variable
 MAKLMGRD_LICENSE_FILE 2-6
 RTI_CONFIG 2-8
example application 1-4

F

FDD file 2-8
FED file 2-8
 FDD file, XML file 1-5
file

FDD 1-5
FED 1-5, 2-8
RID 2-8
XML 1-5
FLEXlm 2-4
FOM 2-7
 agility 1-6

H

HLA, specification 1-5
hostname, license server 2-5

I

IEEE, 1516 API 1-5
ifconfig 2-10
installing, RTI 2-8

L

License Manager 2-4
license server, hostname 2-5
License Setup dialog box 2-5
Linux, installing on 2-2

M

MÄK RTI, configuring 2-8
MAKLMGRD_LICENSE_FILE, environment
 variable 2-6
manual, location of 2-3

N

netmask, configuring 2-9

P

path, specifying for RTI 2-8

port 2-9

 default 2-10

protocol, supported 1-4

protocol-independent, API 1-2

R

Real- Time Platform Reference FOM 1-6

 See also RPR FOM

RID file 2-8

rid.mtl file 2-8

RPR FOM 1-6

RTI 2-7

 1.3

 compatibility with 1516 1-5

 1516

 compatibility with 1.3 1-5

 definition of 2-7

RTI_CONFIG environment variable 2-8

S

sample applications 1-4

simulation, standard 1-4

SISO DLC HLA API 1516 1-5

SISO-STD-004.1-2004 1-5

specification, HLA 1-5

standards, simulation 1-4

supported protocols 1-4

U

UDP, port 2-9–2-10

UNIX, broadcast address 2-10

V

VR-Link

 defined 1-1

 files 2-3

W

Windows, installing on 2-2

X-Y-Z

XML, RTI configuration file 2-8



Link - Simulate - Visualize

VRL-4.0.6-4-121031