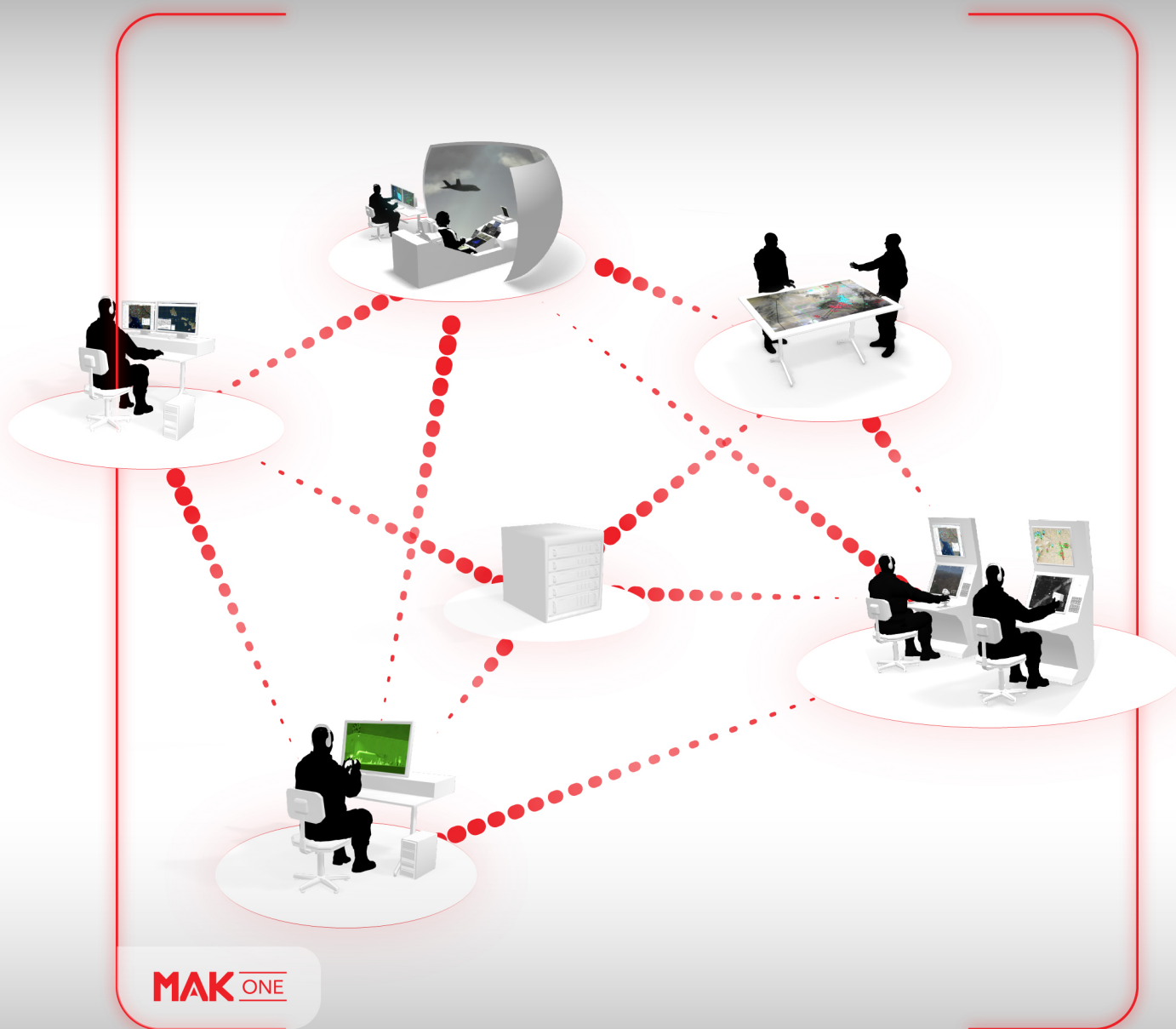




Copyright © 2024 MAK Technologies
All rights Reserved. Printed in the United States.

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

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

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

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

SilverLining™ is a trademark of Sundog Software.

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

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

All other trademarks are owned by their respective companies.

For third-party license information, see "Third-Party Licenses" on page 10 in the Preface.

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

Voice: +1 617.876.8085

Fax: +1 617.876.9208

info@mak.com

www.mak.com

Revision VRL-5.9.1-4-240819



Contents

Preface

How This Manual Is Organized	6
VR-Link Documentation Set	6
MAK Products	7
How to Contact Us	9
Telephone	9
E-mail	9
Internet	9
Post	9
Document Conventions	9
Mouse Button Naming Conventions	10
Third-Party Licenses	10
Third-Party Libraries Supported	10
Library Licensing	11
Boost License	13
libXML and libICONV	14

1. Introduction

1.1. The Protocol-Independent API	16
1.2. VR-Link Features	16
1.2.1 Files and Executables in VR-Link	17
1.3. Simulation Standards Supported	18
1.3.1 Support for the HLA RTI 1.3 and RTI 1516 Specifications	18
1.3.2 HLA FOM Support	19
1.4. Developer Documentation	19

2. Installing and Configuring VR-Link

2.1. Installing VR-Link	22
2.1.1 Installation and Data Packages	22
2.1.2 Installing VR-Link on Windows	22
2.1.3 Installing VR-Link on Linux Systems	23
2.1.4 Installing VR-Link PDB Files for Debugging Purposes	24
2.2. The VR-Link Directory Structure	25
2.3. Installing and Setting Up the MAK License Manager	26
2.3.1 Specifying the License Server	26
2.4. Installing an RTI	28
2.4.1 Installing the MAK RTI	29
2.5. Configuring the Network for DIS Applications	30
2.5.1 Broadcast Address and Netmask (UNIX)	30
2.5.2 UDP Port	30
2.6. Disabling and Enabling Dark Mode	30

A. Troubleshooting

A.1. The Fault Tolerant Heap	32
A.2. Backward Compatibility	32
A.2.1 Using the Compatibility Directory	32

MAK Products Glossary 33

Index 45



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" on page 15 describes the main features of VR-Link and contains information about simulation standards.

"Chapter 2. Installing and Configuring VR-Link" on page 21 explains how to install VR-Link and other required and optional software.

"Appendix A. Troubleshooting" on page 31 explains how to resolve some common problems that can arise with VR-Link.

The "MAK Products Glossary" on page 33 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 *vrlink5.9.1/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, we recommend the free Acrobat Reader; you can get it from Adobe at <https://get.adobe.com/reader>.

MAK Products

The MAK Technologies product line includes:

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

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

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

- **VR-Vantage™.** VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes VR-Vantage and the VR-Vantage Toolkit.
 - VR-Vantage displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to simulation objects so that it moves as they do.
 - **SensorFX.** SensorFX is an enhanced version of VR-Vantage that uses physics-based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
 - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK's VR-Vantage applications, or to integrate VR-

Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.

- **MAK Data Logger.** The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- **VR-Exchange™.** VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- **VR-TheWorld™.** VR-TheWorld is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- **DI-Guy™.** DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- **RadarFX.** RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.
- **VR-Engage.** VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- **WebLVC Server.** WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.
- **MAK Legion.** MAK Legion is a collection of applications and an SDK designed to support simulations with millions of simulation objects.

How to Contact Us

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

Telephone

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

E-mail

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

Internet

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

Post

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

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

Document Conventions

This manual uses these typographic conventions:

Sans Serif Bold	Indicates a key on the keyboard or menu options. Mentions of letter keys do not require you to press the Shift key. If Shift is necessary, that will be included, for example, Shift+Q .
	Also indicates parameters.
Monospaced	Indicates values you enter such as commands or arguments.
<i>Monospaced</i>	Indicates command variables that you replace with appropriate values.
<i>Italic</i>	
{ }	Indicates required arguments.
[]	Indicates optional arguments.
	Separates options in a command where only one option may be chosen at a time.
()	In command syntax, indicates equivalent alternatives for a command-line option, for example, (-h --help).

/	Indicates a directory. Since MAK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames.
<i>Italic</i>	Indicates a file name, pathname, or a class name.
►	Indicates a procedure.
Menu > Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose File > Save .
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the VR-Link home directory. For example, the directory `./vrlink5.9.1/doc` appears in the text as `./doc`.

Mouse Button Naming Conventions

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

Third-Party Licenses

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

Third-Party Libraries Supported

VR-Link 5.9.1 uses the indicated versions of these libraries:

- VR-Link 5.9.1
- Qt 5.15.2 (VC 15 and Red Hat 8)
- OpenSceneGraph 3.6.4
- Boost 1.63.0
- FreeGlut 2.4.0 (Windows), 2.6.0 (Linux)
- GL Studio 8.2
- DI-Guy 13.7 (development version)
- Triton 4.43
- SilverLining 6.20
- SpeedTree 6.3.2
- CDB API 2.2 (Windows), 2.1 (Linux)

- osgEarth 3.2
- CUDA 11.7

Library Licensing

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

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

These libraries are covered by the ZLib license:

- Bullet Physics
- LibPNG
- LibSDL
- Minizip
- Zlib

These libraries are covered by the MIT license:

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

These libraries are covered by the BSD license:

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

These libraries are covered by the commercial licenses:

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

These libraries are covered by custom licenses:

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

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLite is in the public domain.

SWIG is covered by the GPL license.

Boost License

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

Boost Software License - Version 1.63 - August 17th, 2003

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

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

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE

LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link, and all MAK ONE products that use VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MAK products. MAK has made no modifications to these libraries. For more information about these libraries, see the following websites:

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



1. Introduction

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.

1.1. The Protocol-Independent API	16
1.2. VR-Link Features	16
1.2.1 Files and Executables in VR-Link	17
1.3. Simulation Standards Supported	18
1.3.1 Support for the HLA RTI 1.3 and RTI 1516 Specifications	18
1.3.2 HLA FOM Support	19
1.4. Developer Documentation	19

1.1. The Protocol-Independent API

With VR-Link's protocol-independent APIs, 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, and provides notice when an entity enters or leaves an exercise. It can also perform 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.

¹The 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, you do not need to change code that uses the API 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.
- A C# API and a Java API provide additional flexibility for creating VR-Link applications.
- 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 has 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, as well as executables and source code for these utility applications:

- *f18*
- *netdump*
- *talk*
- *listen*

VR-Link also includes source code and executables for other sample applications. The "test" applications demonstrate how you can extend the features of VR-Link by creating your own PDUs, interactions, encoders, and decoders. For more information on these sample applications, refer to "Example and Utility Applications" in the VR-Link Class Documentation.

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, refer to the [Product Specifications & Dependencies](https://www.mak.com/mak-one-specs) page (<https://www.mak.com/mak-one-specs>) and select VR-Link.

For a brief discussion of DIS and HLA, 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 MAK 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, see Section 5.11, "Interoperability Between HLA 1.3 and IEEE 1516 Federates," in the API documentation.

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

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 named *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.

1.3.2 HLA FOM Support

VR-Link provides extensive built-in support for the Real-Time Platform Reference FOM (RPR FOM). (For version information, see the Release Notes.) 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 MAK Technologies, support this FOM.

VR-Link supports portions of the NATO Education and Training Network Distributed Simulation Federation Object Model (NETN FOM) for HLA Evolved and DIS, specifically the Physical and METOC FOM Modules of the NETN FOM. The NETN FOM is an extension of the RPR FOM 2.0 and adds extended object and interaction classes for various attributes and data. VR-Link defaults to using the NETN FOM for HLA Evolved applications. DIS supports NETN attributes and objects through custom attribute records and PDUs.

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, see Section 6, FOM Agility, in the API documentation.

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

1.4. Developer Documentation

You can access documentation of the VR-Link APIs as a set of HTML files by starting at *./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.



2. Installing and Configuring VR-Link

This chapter covers:

2.1. Installing VR-Link	22
2.1.1 Installation and Data Packages	22
2.1.2 Installing VR-Link on Windows	22
2.1.3 Installing VR-Link on Linux Systems	23
2.1.4 Installing VR-Link PDB Files for Debugging Purposes	24
2.2. The VR-Link Directory Structure	25
2.3. Installing and Setting Up the MAK License Manager	26
2.3.1 Specifying the License Server	26
2.4. Installing an RTI	28
2.4.1 Installing the MAK RTI	29
2.5. Configuring the Network for DIS Applications	30
2.5.1 Broadcast Address and Netmask (UNIX)	30
2.5.2 UDP Port	30
2.6. Disabling and Enabling Dark Mode	30

2.1. Installing VR-Link

2.1.1 Installation and Data Packages

MAK ONE installations are provided in one or more files (packages) that you can download from the MAK website.

There are four types of installation products for MAK ONE:

- Product installers – Each product has a single main installer for each operating system.
- Optional installers – Some products have installers for developer options.
- Required data packages – MAK ONE applications (VR-Vantage, VR-Forces, VR-Engage, SensorFX, and RadarFX) and the DI-Guy SDK have a collection of data packages that are required for those products' basic functions.
- Recommended data packages – Additional data packages (if available) are recommended to access the full capabilities of MAK ONE applications.

As of spring 2022, new releases of MAK ONE applications share the same data installation, therefore the same set of data packages is all you need for any or all applications (VR-Vantage, VR-Forces, VR-Engage, SensorFX, and RadarFX).

Dependencies

Licensing tools and dependent products have their own installers. For details, see "2.3. Installing and Setting Up the MAK License Manager" on page 26.

For interoperability using HLA, see "2.4. Installing an RTI" on page 28.

Before you install , please read the release notes to see if there are any special instructions for installation.

2.1.2 Installing VR-Link on Windows

Windows versions of VR-Link are provided as downloaded files.

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

Note the following:

- The installers are named to indicate the compiler used to build that version of VR-Link.
- When you install large applications on Windows, there may be a delay of up to several minutes from the time you try to run the setup program to the time that an installation dialog box displays. 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.

Default Windows Installation Directory

The default root directory for installing MAK ONE products is *C:/MAK*.

You can use an environment variable, `MAK_INSTALL_DIR`, to specify a different location where all MAK ONE products and data should be installed by default. For example, if you define it as `MAK_INSTALL_DIR = D:\MyProject`, then all MAK ONE products automatically install to that location. The data packages and products would then install in that directory by default: `D:/MyProject/SharedData`, `D:/MyProject/vrlink5.9.1`, and so on.

Uninstalling from Windows

MAK ONE product installations include an uninstaller application (*unins000.exe*), which you can find in the installation directory. Run that program to uninstall the product.

When you run other installers for the product, such as the PDBs, there are additional uninstallers. If you run the uninstallers from the highest number down (for example, *unins002.exe* and then *unins001.exe*), running *unins000.exe* uninstalls everything else and removes the install folder.

When you uninstall VR-Vantage, VR-Forces, or VR-Engage, files such as scenarios or SMSs that reside in the product directory are also deleted. Before uninstalling, copy any of your custom files from the product directory to another folder.

The uninstaller does not remove the data package. That is in a separate directory (for example, `./SharedData/##/latest`).

i If you run only the product uninstaller, you can then simply delete the rest of the product's install folder. Note that if you do not run all of the uninstallers, running only the product uninstaller may not remove all of the shortcuts from the Windows Start menu.

2.1.3 Installing VR-Link on Linux Systems

Linux versions of VR-Link are provided as compressed TAR files.

► To install a product from a TAR file:

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

```
tar -xzf applicationxx-os.tar.gz
```

where *xx-os* is the release, operating system, and other product release coding.

Uninstalling from Linux

To remove installations from Linux machines, simply delete the installation directory. This catches all of the files associated with the product.

When you uninstall VR-Vantage, VR-Forces, or VR-Engage, files such as scenarios or SMSs that reside in the product directory are also deleted. Before uninstalling, copy any of your custom files from the product directory to another folder.

Deleting the installation directory does not remove the data package. That is in a separate directory (for example, `./SharedData/##/latest`).

2.1.4 Installing VR-Link PDB Files for Debugging Purposes

When a MAK ONE application crashes, it can be difficult to determine the cause of the crash. The application generates a *.dmp* file that contains valuable information about the state of the application at the time of the crash. MAK technical support can analyze this file to help diagnose the issue.

In addition, with the application's PDB (program database) files, the application is able to generate a call stack log file (*.callstack.log*) of the application at the time of the crash. The call stack log provides information in plain text about the sequence of functions that were called leading up to the crash, which is also helpful in identifying the specific cause.

By providing the *.dmp* or *.callstack.log* files to technical support, you help support personnel to resolve the issues. They use specialized debugging tools to analyze the files and determine the cause of the crash, and that information aids in the development of a solution to prevent future crashes.

Due to security reasons or organization policies, some customers cannot send the *.dmp* file to technical support. Because the call stack log file is a plain text file, it is easier to inspect and analyze than the *.dmp* file. Users can easily attach it to an email and pass it out of secure areas with visual inspection, making it a convenient option for customers with restrictions on sharing binary files.

It is important to note that the call stack log file alone might not be sufficient for support personnel to fully diagnose the issue. They may require more information to gain a complete understanding of the issue. A detailed procedure for reproducing the crash often helps them solve the problem you encountered.



- Support for the generation of the call stack log file was introduced in MAK ONE 2024. It is supported only in MAK ONE products part of that release and later versions.
- The PDB files are part of the Linux installation. You do not need to install them separately.

Installing PDB Files on Windows

Before installing the PDB files on Windows, you must install VR-Link. Next, make sure that the PDB files you are installing are from the same build as the application. When you download the installer from the [MAK ONE Product Downloads](#) page, the appropriate PDB installer is also available for download.

► **To install the PDB files:**

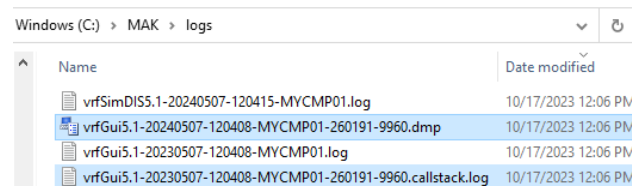
1. After installing VR-Link, run the PDB installer for that product.
2. The installer looks for the matching product directory by searching the registry and then it automatically sets the install directory to the appropriate path. If that is not correct for your product, see the note below.
3. Complete the installation wizard. The necessary PDB files are installed in the product's *./bin64* directory.

- The default installation directory is `C:\MAK`. See "Default Windows Installation Directory" on page 22 to learn about setting the default to other locations.
- Be sure that the folder in the Select Destination Location page of the installation wizard is correct. It should be the top-level folder for the product installation.
- If the PDB installer displays a warning that the SVN revisions do not match, make sure that you have the correct PDB installer for the version of VR-Link that you installed.

Reporting an Issue Using the `.callstack.log` and `.dmp` Files

Send the `.callstack.log` and `.dmp` files (or only the `.callstack.log` if you are unable to provide the `.dmp` file) to MAK support through the Support Portal (<https://www.mak.com/support>) or support@mak.com. The files will have similar names and are located in the central directory for MAK ONE log files (Figure 1). Please also provide information on anything you experienced when the issue occurred.

Figure 1. Example of `.callstack.log` and `.dmp` files for VR-Forces



- The default directory for MAK ONE log files is `C:/MAK/logs`. You can customize the location by specifying a different path in the `MAK_LOG_DIR` environment variable.

2.2. The VR-Link Directory Structure

The installation process creates a directory structure like the one shown in Figure 2. On PCs, the default installation directory is `C:/MAK/vrlink#.#`.

Figure 2. VR-Link directory structure

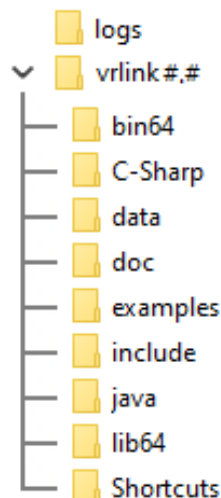


Table 1 describes the contents of the subdirectories.

Table 1. Contents of VR-Link subdirectories

Directory	Directory contents
<code>./bin64</code>	Executables, including utilities and example applications.
<code>./C-Sharp</code>	C# examples and Code Generator executables.
<code>./data</code>	Miscellaneous data files that are not appropriate for the bin directories.
<code>./doc</code>	VR-Link documentation in PDF format.
<code>./examples</code>	Source directories for the example applications.
<code>./include</code>	C++ header files.
<code>./java</code>	Java API files and examples.
<code>./lib64</code>	VR-Link C++ libraries.
<code>./Shortcuts</code>	Contains the shortcuts to the documentation and tools, which are used for the Windows Start menu.

2.3. Installing and Setting Up the MAK License Manager

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

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

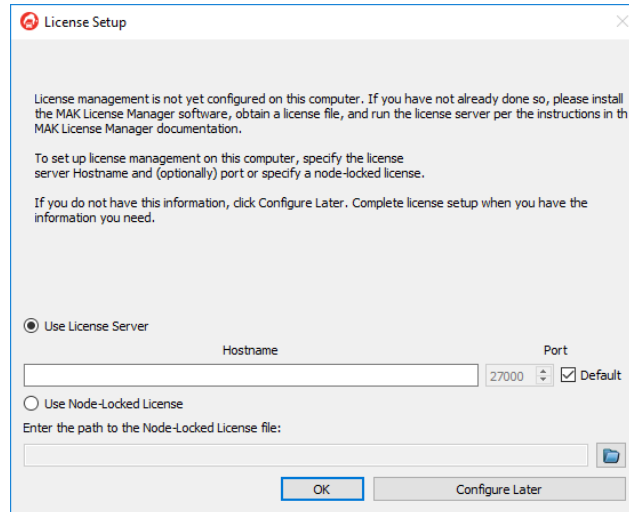
The License Manager installer is included with MAK installation media. It is separate from the product installers. You can download the installers from the [MAK ONE Product Downloads](https://www.mak.com/mak-one-downloads) page at <https://www.mak.com/mak-one-downloads>.

Complete installation and configuration instructions are included with the License Manager installer. The *MAK ONE Installation Guide* is available at <https://mak.com/mak-one-installation-guide>.

Some customers use dongle licenses instead of running a license server. Instructions for using dongles are also in the *MAK ONE Installation Guide*.

2.3.1 Specifying the License Server

The first time you run a MAK ONE product on a particular computer, the License Setup dialog box opens (Figure 3). It prompts you to enter the hostname of the license server and, optionally, a port number.

Figure 3. License Setup dialog box

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

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

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



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

Setting 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 or node-locked license 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.

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

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

Windows

► To set MAKLMGRD_LICENSE_FILE in Windows:

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

Linux

On Linux, you set environment variables in your `.cshrc` (or equivalent startup file).

► To specify a license server, set the variable similarly to:

```
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:

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

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

You are ready to run the license server and use your new licenses or MAK ONE 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. MAK Technologies has the MAK RTI, but MAK ONE products do not require that you use the MAK 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 MAK ONE products, please visit the Product Specifications & Dependencies page (<https://www.mak.com/mak-one-specs>).

The following instructions explain how to install the MAK RTI. If you are using an RTI provided by another vendor, please refer to that RTI's documentation for installation and setup instructions.

2.4.1 Installing the MAK RTI

To install the MAK RTI, follow the instructions in Chapter 2 of *MAK RTI Users Guide*. To install another RTI, see the vendor's documentation.

Configuring Your System to Use the MAK RTI

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

- Linux — LD_LIBRARY_PATH.
- Windows — PATH.
- The MAK RTI needs to know where to find these configuration files:
 - The federation configuration file (*.fed*, *.fdd*, or *.xml*) for your federation execution (required).
 - The RID file (*rid.mtl*) (optional).

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

Running Applications with the MAK RTI

► To run a MAK ONE product with the MAK RTI:

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

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

2.5. 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 uses the broadcast address of the first device listed in the interface table by default.

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 MAK applications allow you to set the port number with the `-P port` option or in a connection dialog box.

2.6. Disabling and Enabling Dark Mode

Dark mode is the default in VR-Link Code Generator's graphical user interface. Light mode, the former standard, is also available.

- To switch the GUI between dark and light mode, choose **Configuration > Toggle Dark Mode**.



A. Troubleshooting

This appendix describes solutions to common issues. If your problem is not covered here, visit MAK's [Support Portal](https://www.mak.com/support) (<https://www.mak.com/support>) to request assistance.

A.1. The Fault Tolerant Heap	32
A.2. Backward Compatibility	32
A.2.1 Using the Compatibility Directory	32

A.1. The Fault Tolerant Heap

The Fault Tolerant Heap (FTH) is a subsystem on Windows that can affect software developers and testers. If a software application crashes many times—for example, in the process of programming and debugging that application—Windows adds it to the FTH. This is significantly slower than the normal heap and can have a negative impact on the performance of the application as a result.

For more information and to learn possible solutions, see <https://docs.microsoft.com/en-us/windows/win32/win7appqual/fault-tolerant-heap>.

A.2. Backward Compatibility

As of VR-Link 5.9, there are some backwards compatibility issues:

- All HLA federates now create and join a federation named "MAK-One-YYYY" by default, while previous versions of VR-Link used a "MAK-RPR-2.0" federation name.
- HLA 1516 Evolved federates use the NETN FOM and NETN FOM Mapper by default. HLA 1516 Evolved federates publish extended entity classes derived from the RPR 2.0 FOM and NETN FOM. Older VR-Link federates do not support these new attributes, but they are still able to subscribe and receive RPR 2.0 objects and attributes.
- The default FOM Configuration has changed for HLA Evolved Federates. Joining federations with previous VR-Link federates will result in FOM merge errors unless steps are taken to ensure federates use the same FOM.
- VR-Link 5.9 added DIS IFF interactive mode support. This can cause issues with older VR-Link applications that attempt to decode these IFF PDUs.

A.2.1 Using the Compatibility Directory

VR-Link has a *compatibility* directory in its *include* directory. This directory has the same directory structure as the main *include* directory, with header files that have the same names as previous versions of VR-Link. The header files in this directory do not duplicate the declarations in the renamed header files; they include the renamed header files themselves. To use the *./include/compatibility* directory, add it to your project's include directory list. You will not need to update the include statements in your project files.



MAK Products Glossary

This glossary defines terms used in MAK ONE product documentation.

A**absolute dead reckoning**

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

aggregate

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

API

Application Programming Interface.

articulated part

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

attached part

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

B**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.

C**channel**

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

clipping

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

clipping planes

The range of distances from the observer (near clipping and far clipping) in which an application clips (does not display) 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.

D**Data Distribution Management**

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

dead-reckoning

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

DDM

See Data Distribution Management.

Defense Modeling and Simulation Office

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

DIS

See Distributed Interactive Simulation.

DIS data packets

See Protocol Data Unit.

display engine

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

Distributed Interactive Simulation

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

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

DMSO

See Defense Modeling and Simulation Office.

E**element definition**

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

emitter

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

entity

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

entity maintenance

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

entity-type

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

The components of an entity-type are:

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

environmental process

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

Euler angles

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

event

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

exercise

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

exercise connection

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

exercise ID

A numeric identification for a DIS simulation exercise.

F**FED file**

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

federate

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

federation

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

Federation Object Model

Defines the data content of a federation execution.

federation execution

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

field of view

Controls the perspective of the observer.

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

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

filter range

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

FOM

See Federation Object Model.

FOM Module

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

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

frame rate

The rate at which the application displays updated images.

G**ground clamping**

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

geocentric coordinates

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

geodetic coordinates

A coordinate system in which position is determined relative to a reference ellipsoid, such as the surface of the earth at sea level. In MAK ONE 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.

H**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 Protocol Data Units are sent to the network regardless of whether the entity's state has changed.

High-Level Architecture

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

HLA

See High-Level Architecture.

I**IG**

image generator

interaction

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

L**Logger control PDUs**

A set of DIS PDUs or HLA interactions that let you control the MAK Data 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 a Run-Time Infrastructure. A federate communicates with the HLA network through the LRC.

LRC

See Local RTI Component.

M**MÄK Technologies Lisp**

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

message

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

MIM

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

model definition

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

Modeling and Simulation Coordination Office

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

MSCO

Modeling and Simulation Coordination Office. (Formerly DMSO.)

MTL

See MAK Technologies Lisp.

N**NETN FOM**

The NETN FOM (NATO Education and Training Network Distributed Simulation Federation Object Model) is an extension of the RPR FOM 2.0, allowing other federates using pure RPR FOM to interoperate with applications using the NETN FOM.

O**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 Run-Time Infrastructure 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 Run-Time Infrastructure, and the RTI provides functions to find out an object's name, given its handle, and vice versa. Object names can be chosen by applications that register the objects with the RTI, however if you do not want to choose names for objects, the RTI will assign names for you.

observer

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

P**PDU**

See Protocol Data Unit.

Protocol Data Unit

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

R**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.

Real-time Platform Reference FOM

An HLA reference Federation Object Model based on the DIS protocol.

recording

A MAK 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 Federation Object Model 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. Contrast with absolute dead reckoning.

remote display engine

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

RID file

See RTI Initialization Data.

RTI Initialization Data

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

RPR FOM

See MAK Products Glossary.

RTI

See Run-Time Infrastructure.

Run-Time Infrastructure

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

S**scene element**

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

Simulation Interoperability Standards Organization

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

simulation time

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

SISO

See Simulation Interoperability Standards Organization.

smooth period

The period of time over which trajectory smoothing takes place.

smoothing

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

state

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

T**tape**

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

terrain following

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

timeout

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

timescale

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

topographic coordinates

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

topographic coordinate frame

A coordinate frame in the context of the terrain.

trajectory smoothing

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

U**UDP port**

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

Universal Transverse Mercator

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

UTM

See Universal Transverse Mercator.



view control messages

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



Index

A

- address, broadcast, 30
- API documentation, 19
- application, example, 17

C

- C# API, 17
- callstack.log file, 24
- compatibility
 - 1.3 and 1516 APIs, 18
- configuration
 - dark or light mode, 30
 - MAK RTI file, 29
- configuring
 - broadcast address and netmask for UNIX, 30
 - netmask, 30
 - VR-Link for DIS, 30
- crash, application, 25

D

- dark mode, 30
- data
 - install, 22
- debugging
 - callstack.log file, 24
 - dmp file, 24
 - file location, 25
- default port, 30
- developer documentation, 19

- directory
 - tree, VR-Link, 25
 - Windows default install, 22
- DIS, network configuration, 30
- dmp file
 - debugging, 24
 - location, 25

E

- environment variable
 - MAK_INSTALL_DIR, 22
 - MAKLMGRD_LICENSE_FILE, 27
 - RTI_CONFIG, 29
- example application, 17
- executable
 - installation location
 - Windows, 22

F

- fault tolerant heap, disabling, 32
- FDD file, 19, 29
- FED file, 19, 29
- file
 - FDD, 19, 29
 - FED, 19, 29
 - RID, 29
 - XML, 19
- FLEXIm, 26
- FOM, 28
 - agility, 19
- FTH, 32

H

heap, fault tolerant, 32
HLA specification, 18
hostname
 license server, 26

I

IEEE
 1516 API, 18
ifconfig, 30
installation, 22
 default directory, Windows, 22
 Linux, 23
 RTI, 29
 Windows, 22

J

Java API, 17

L

license manager, 26
license server
 hostname, 26
Linux
 product installation, 23
log file
 callstack, 24

M

MAK RTI configuration, 29
MAK_INSTALL_DIR environment variable, 22
MAKLMGRD_LICENSE_FILE environment variable,
 27

N

netmask, configuring, 30

P

path
 default Windows install, 22
 MAK RTI, 29

PDB files
 Linux, 24
 Windows, 24
performance
 fault tolerant heap, 32
port, default UDP, 30
port, UDP, 30
product installation, 22
 Linux, 23
 Windows, 22
protocol, supported, 18
protocol-independent API, 16

R

Real-Time Platform Reference FOM, See RPR FOM
RID file, 29
rid.mtl file, 29
RPR FOM, 19
RTI, 28
 1.3 and 1516 APIs compatible, 18
RTI_CONFIG environment variable, 29

S

sample applications, 17
simulation, standard, 18
SISO DLC HLA API 1516, 18
SISO-STD-004.1-2004, 18
specification, HLA, 18
standards, simulation, 18
supported protocols, 18

T

troubleshooting, 32

U

UDP port, 30
UNIX, broadcast address, 30

V

variable, environment, 27
VR-Link
 defined, 15
 files, 25



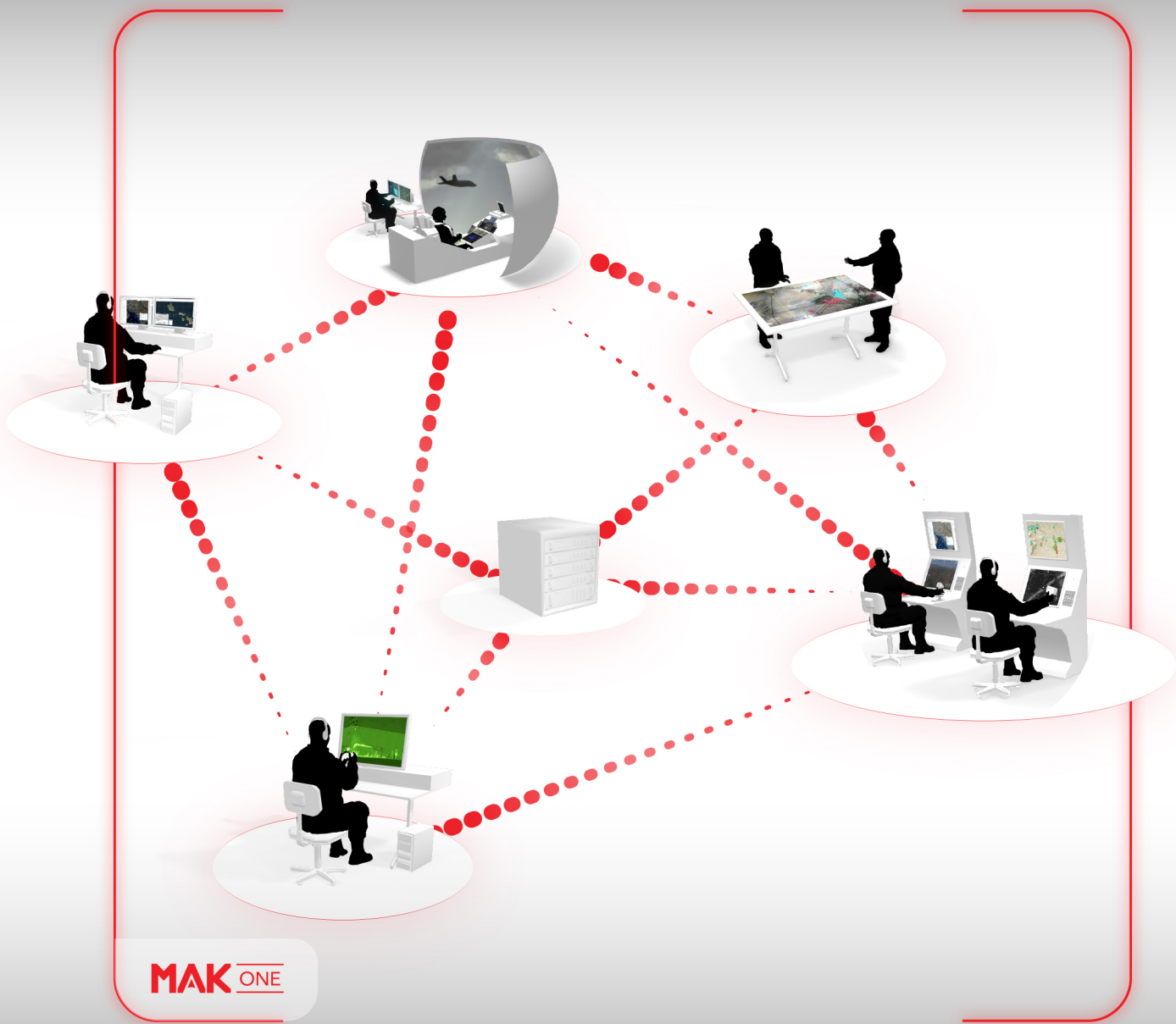
Windows

- default install directory, 22
- product installation, 22



XML

- RTI configuration file, 29



VRL-5.9.1-4-240819