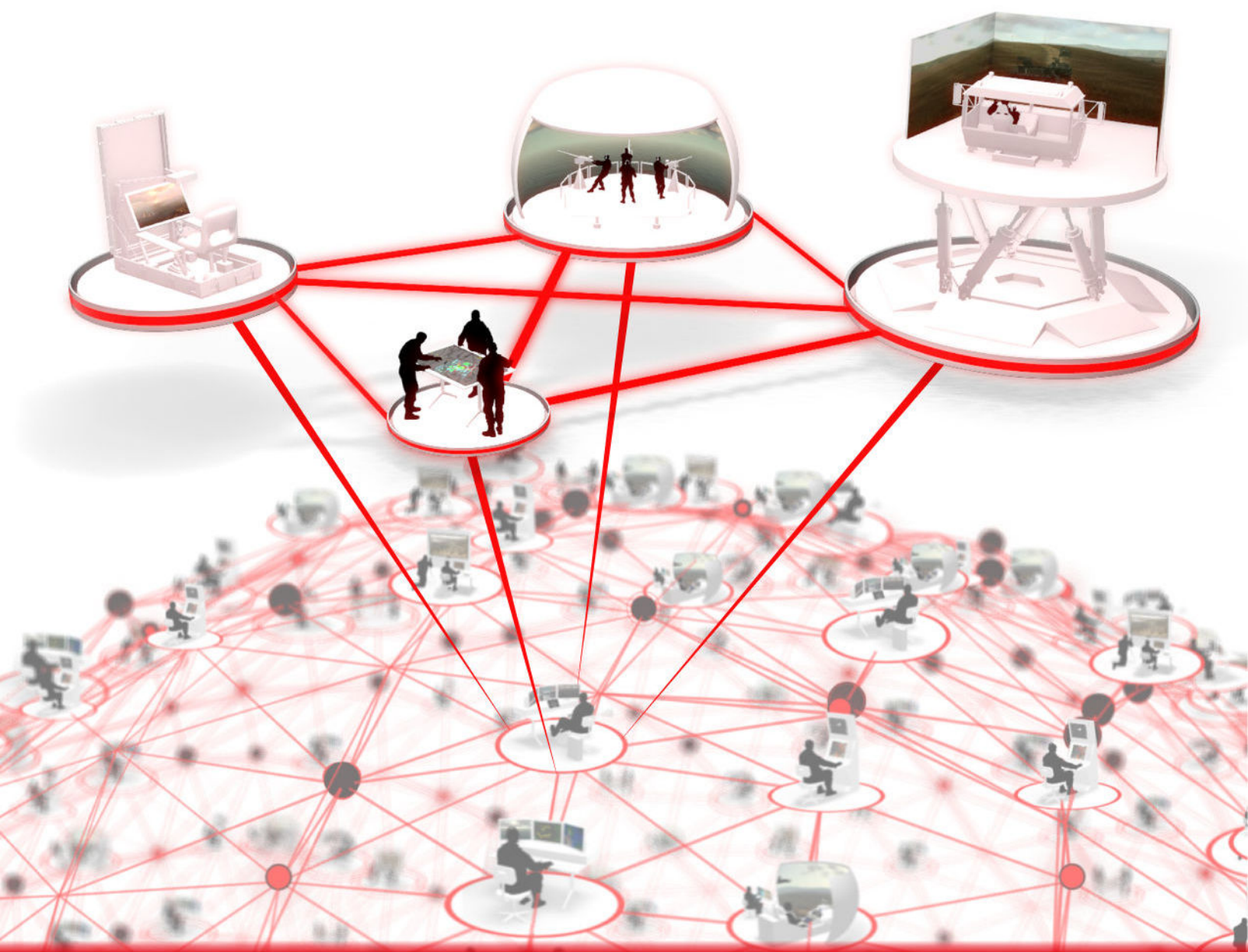




Copyright © 2025 MAK Technologies
All rights Reserved. Printed in the United States.

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.

RFView® is a registered trademark of Information Systems Laboratories (ISL), Inc.

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, visit the MAK ONE Product Support Knowledge Base (at <https://www.mak.com/knowledge>) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).

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 MAK-25.0-03-251024



Contents

1. MAK Product Interoperability

1.1. The MAK Technologies Product Line	8
1.2. Basic Requirements for Running Any MAK ONE Product	10
1.2.1 Running the License Server	10
1.3. Running HLA Applications	10
1.4. Running DIS Applications	11
1.5. General Issues for Product Compatibility	11
1.5.1 Operating System Compatibility	12
1.5.2 Development Environment Compatibility	12
1.5.3 Third-Party API Compatibility	12
1.5.4 License Management Compatibility	13
1.6. Intra-Product Compatibility	13
1.6.1 VR-Link	13
1.6.2 MAK RTI	13
1.6.3 VR-Forces	13
1.6.4 VR-Engage	13
1.6.5 VR-Vantage	14
1.6.6 MAK Data Logger	14
1.6.7 VR-Exchange	15
1.6.8 WebLVC Server	15
1.6.9 VR-TheWorld	15
1.7. Inter-Product Compatibility	16
1.7.1 VR-Engage	16
1.7.2 Recording VR-Forces Simulations with the MAK Data Logger	16
1.7.3 FED File Naming Conventions	17
1.7.4 Compatibility Tables	19
1.8. The Exercise Connection Configuration File	20
1.8.1 Exercise Configuration File Parameters	20

1.8.2 Specifying Exercise Connection Configuration Files on the Command Line	20
1.9. Synchronizing Terrain for the VR-Vantage and VR-Forces	21
1.9.1 Flat Earth Databases	21
1.9.2 Unsupported Features	21
1.9.3 Database is not WGS84	21
1.9.4 Optimized Databases	22
1.10. Delivering Applications to Users	22

2. The MAK Object Model

2.1. MAK Object Model Overview	24
2.2. DIS and RPR FOM Support	25
2.3. NETN FOM Support	27
2.4. MAK Object Model Extensions	29
2.4.1 Aerodromes and Runways	29
2.4.2 Aggregates	33
2.4.3 Dynamic Terrain	33
2.4.4 Human Characters (DI-Guy)	38
2.4.5 IFF	39
2.4.6 Logger Control	39
2.4.7 Physical Entities	39
2.4.8 Radar	46
2.4.9 Sensor Field of View	47
2.4.10 Time and Date	49
2.4.11 Weather	51

3. VR-Forces External Communication Server Interface

3.1. Overview	64
3.1.1 Messages From VR-Forces Sim	64
3.1.2 Messages To VR-Forces Sim	64
3.1.3 Ready To Send Mode	64
3.2. Request Message	65
3.2.1 Signal Data String	65
3.3. Response Messages	66
3.3.1 Process Message	66
3.3.2 Timeout	67
3.4. Ready To Send Signal (RTSS)	67

4. Default Ports for MAK ONE Products

4.1. MAK Legion	70
4.2. MAK Data Logger	70
4.3. MAK RTI	70

4.4. VR-Link	70
4.5. VR-Exchange	71
4.6. WebLVC	71
4.7. VR-Vantage	71
4.8. VR-Forces	71
4.9. VR-Engage	72
4.10. RFView for MAK ONE	72
4.11. RadarFX	72
4.12. FLEXIm Licensing	73
4.13. Tracy	73
4.14. Loopback Connections	73

5. Basic Troubleshooting

5.1. Product Does Not Run	76
5.1.1 Linux Applications	76
5.1.2 HLA Applications	76
5.1.3 License Management Checklist	77
5.2. MAK ONE Products Do Not Work Together	79
5.2.1 DIS Applications	79
5.2.2 HLA Applications	81
5.3. Using Custom FOM Configurations	84
5.4. Using the DIS Spy	84
5.4.1 Starting the DIS Spy	84
5.4.2 Connecting to a Simulation	84
5.4.3 Viewing PDU Information	84
5.5. Using Older MAK ONE Products with HLA Evolved	85
How to Contact Us	85
Telephone	85
E-mail	85
Internet	85
Post	86

6. Conventions and Third-Party Licenses

Document Conventions	88
DI-Guy Conventions	88
Mouse Button Naming Conventions	89
Third-Party Licenses	89

Index	91
--------------------	-----------



1. MAK Product Interoperability

This handbook provides a high-level introduction to using MAK ONE products in DIS and HLA exercises. MAK ONE products are fully interoperable with other applications that adhere to the DIS or HLA standards. In order to interoperate, however, you must properly configure your applications. This handbook describes the basic requirements to run our products using these simulation standards and discusses compatibility issues among different versions of MAK ONE products. We hope that it will help both new and experienced customers to set up their simulation environments. Chapter 3 provides basic troubleshooting procedures.

For specific installation and configuration instructions, please see the documentation for your particular MAK ONE products. You can also contact support at <https://www.mak.com/mak-one/support/portal>.

1.1. The MAK Technologies Product Line	8
1.2. Basic Requirements for Running Any MAK ONE Product	10
1.2.1 Running the License Server	10
1.3. Running HLA Applications	10
1.4. Running DIS Applications	11
1.5. General Issues for Product Compatibility	11
1.5.1 Operating System Compatibility	12
1.5.2 Development Environment Compatibility	12
1.5.3 Third-Party API Compatibility	12
1.5.4 License Management Compatibility	13
1.6. Intra-Product Compatibility	13
1.6.1 VR-Link	13
1.6.2 MAK RTI	13
1.6.3 VR-Forces	13
1.6.4 VR-Engage	13
1.6.5 VR-Vantage	14
1.6.6 MAK Data Logger	14
1.6.7 VR-Exchange	15
1.6.8 WebLVC Server	15

1.6.9 VR-TheWorld	15
1.7. Inter-Product Compatibility	16
1.7.1 VR-Engage	16
1.7.2 Recording VR-Forces Simulations with the MAK Data Logger	16
1.7.3 FED File Naming Conventions	17
1.7.4 Compatibility Tables	19
1.8. The Exercise Connection Configuration File	20
1.8.1 Exercise Configuration File Parameters	20
1.8.2 Specifying Exercise Connection Configuration Files on the Command Line ..	20
1.9. Synchronizing Terrain for the VR-Vantage and VR-Forces	21
1.9.1 Flat Earth Databases	21
1.9.2 Unsupported Features	21
1.9.3 Database is not WGS84	21
1.9.4 Optimized Databases	22
1.10. Delivering Applications to Users	22

1.1. The MAK Technologies Product Line

The MAK Technologies line of software products is designed to streamline the process of developing and using networked simulated environments.

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.

- **RFView® for MAK ONE.** Synthetic Aperture Radar (SAR) imaging is an indispensable tool for today's modern militaries. Developed in partnership with ISL, RFView for MAK ONE acts as a SAR server and delivers simulated SAR images that are both physically accurate and fully correlated with out-the-window views and EO/IR images of MAK ONE synthetic environments. It includes a toolkit that enables you to build a client application to request SAR images from the server.
- **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.

1.2. Basic Requirements for Running Any MAK ONE Product

To run any MAK ONE product, you must have a license for that product, you must properly configure all computers on which you are running MAK ONE software, and the license server must be running. Please see the installation instructions for your MAK ONE software for complete details about setting up the license management environment.

1.2.1 Running the License Server

Although product licenses are specific to operating systems, the license server itself can run on any operating system supported by FLEXlm and for which MAK Technologies provides a *maklmgrd* executable (currently Windows and Linux). In other words, if you have a license to run the Linux version of the MAK Data Logger, you could run the license server on a Windows PC.

 All computers on which you are running MAK software must have a network connection to the license server, unless you are using a dongle.

1.3. Running HLA Applications

To run the HLA versions of MAK ONE products, you must also install and configure an RTI. Except as noted below, all MAK ONE products are link compatible with the MAK RTI and other RTIs that support the HLA Evolved (IEEE 1516-2010) or HLA 4 (IEEE 1516-2025) specifications, and which are built with the same compilers as the MAK ONE product you are using.

If you observe these points, you should have no problem running HLA applications:

- You must install an RTI on each computer that is running an HLA federate. Federates must be able to find the RTI libraries (*.dll* or *.so*.) You usually accomplish this on Windows by adding the path to the RTI's *./bin* (or *./bin64*) directory to the PATH environment variable. On Linux, you add the path to the RTI's *lib* directory LD-LIBRARY_PATH environment variable.
- If you are using the MAK RTI, you must have an RTI license for each federate that is participating in the federation.
- All federates in a federation must use the same RTI, configured in the same way, and using the same FED file (FDD file in HLA 1516), and each federate must be able to find the FED file. The easiest way to ensure that a MAK ONE product can find the FED file is to put it in the directory from which you are running the application.
- Some RTIs require that you run an *rtiexec*, but only run one instance of it.
- If you are using the MAK RTI, you can run in lightweight mode without an *rtiexec*. If you want to use advanced features such as DDM, Time Management, or MOM, you must run the *rtiexec*. Run only one instance of the *rtiexec* per federation. Always run different federations on different UDP ports.
- Most RTIs rely on an external configuration file (*rid.mtl* for the MAK RTI.) When you set up your federates, you must ensure that each federate has access to the configuration file.



- When using HLA 4, you must use MAK RTI 5.0 or later.
- Older versions of MAK ONE applications have restrictions on MAK RTI support. See "5.5. Using Older MAK ONE Products with HLA Evolved" on page 85.

1.4. Running DIS Applications

To run the DIS version of a MAK ONE product out-of-the-box, you do not have to do any special configuration or start any supporting applications (except, of course, the license server.) By default, MAK ONE products listen to UDP broadcast traffic on port 3000. If you do not want your DIS application to interact with other DIS applications on your LAN, change the port you are using. You can specify the port in the application's configuration file or with the *-P* command-line option. (Most product manuals list command-line options in the chapter that explains how to start the product. In addition, VR-Vantage, Data Logger, and VR-Forces have quick reference cards bound inside the back cover that list command-line options.)

1.5. General Issues for Product Compatibility

The Product Specifications & Dependencies page on the MAK website at <https://www.mak.com/mak-one-specs> lists a variety of compatibility information for each release of a MAK ONE product. In this section, we summarize at a slightly higher level the factors that affect product compatibility. This should help you make sense of the product-specific listings.

1.5.1 Operating System Compatibility

Compatibility of MAK ONE products across different versions of supported operating systems varies. Windows versions of our products work on all supported versions of Windows, as listed in the Product Specifications & Dependencies page.

Due to changes in system libraries, Linux versions of MAK ONE products usually run only on the version of Linux on which they were compiled.

1.5.2 Development Environment Compatibility

Windows-based MAK ONE products and examples are built with Microsoft Visual C++. Depending on the product, there may be versions built with VC++ 14.0 or 15.0 compilers. (Older compilers are dropped over time.) Product documentation assumes that you are familiar with the use of your chosen compiler and development environment.

 MAK provides versions of product releases that have been compiled with different versions of Microsoft Visual C++. MAK ONE products built with VC++ 14 and later are binary compatible and will work together. When you run MAK ONE products together on the same computer, for example, the Logger and VR-Vantage or VR-Forces, we strongly recommend that you run versions compiled with the same compiler or binary compatible compilers. MAK RTI HLA 4 binary file names conform to the SISO-REF-075 standard, which includes a compiler marker to specify the compiler version. This prevents HLA applications from loading binary files from non-compatible compilers. The MAK RTI HLA 4 libraries are built using the Video Studio 2017 compiler and use the vc141 compiler marker.

If you are running MAK ONE products on different machines, they do not have to be built with the same compiler. However, remember that if you are using the MAK RTI, you must use the same release on all computers (for example, on computer 1, MAK RTI 4.4 VC 14; on computer 2, MAK RTI 4.4 VC 12).

1.5.3 Third-Party API Compatibility

MAK Technologies uses various third-party APIs to build our products. For example, we use the Qt development toolkit for our graphical user interfaces. The *Preface* in the MAK ONE product's user documentation lists the third-party API requirements for that release. "Third-Party Licenses" on page 89 also lists the requirements for recent releases of MAK ONE products, but you should refer to the product documentation to verify what is supported in the version you are using.

Third-party APIs may be updated more frequently than MAK ONE products are updated. Also, in the interest of product stability and interoperability, MAK may not move to upgraded versions of third-party APIs when they become available.

Although you may be able to build MAK ONE products using more recent versions of third-party libraries, this depends on the third-party developer's efforts to maintain backward compatibility. MAK cannot guarantee that this will be possible. In general, when deciding to move to updated third-party libraries, we try to balance the benefits of new features against the inconvenience to customers that may result from porting to newer software.

1.5.4 License Management Compatibility

When MAK ONE products are built, they are linked to a version of the FLEXlm license management software. To run a MAK ONE product, you must install that either version of FLEXlm or a later release. Check the release notes for each release of a MAK ONE product to find out if there are any changes in the release of FLEXlm being used. You can also check the Product Specifications & Dependencies page on the MAK website, <https://www.mak.com/mak-one-specs>, to see which version of FLEXlm is required for a given release. As of March 2022, most products use FLEXlm 11.17.1.

License files are forward compatible, so you do not have to change your license files if you upgrade your FLEXlm software.

1.6. Intra-Product Compatibility

This section discusses backward and forward compatibility of MAK products and applications built using them.

1.6.1 VR-Link

Different versions of VR-Link maintain network compatibility as long as you use the same FOM.

VR-Link 5.0 renamed header files to conform to MAK file naming conventions.

If you are using HLA Evolved, see the restrictions in the note in "1.3. Running HLA Applications" on page 10.

1.6.2 MAK RTI

Federates built with the MAK RTI's HLA 4 libraries can interoperate with federates built with MAK RTI's HLA 1516 Evolved libraries. For a list of restrictions and behavior of certain services when running in a mixed HLA 4/HLA Evolved federation, see "RTI Interoperability" in the *MAK RTI Reference Manual*.

The MAK RTI is binary compatible. This means that you can update the MAK RTI without recompiling any MAK ONE products.

For more information on using HLA Evolved or HLA 4, see "1.3. Running HLA Applications" on page 10.

1.6.3 VR-Forces

The different versions of VR-Forces® and applications built using the VR-Forces toolkit are generally not compatible. This means, for example, that you cannot control a VR-Forces 5.0.3 simulation engine with a VR-Forces 5.1 GUI. Applications built with different versions of VR-Forces can participate in the same exercise. However, they treat each other as remote simulations.

1.6.4 VR-Engage

Due to changes in the simulation model sets between releases, different releases of VR-Engage cannot interact with each other, except that they will see any entities simulated by the other version as remote entities.

1.6.5 VR-Vantage

There are no compatibility issues for using different releases of VR-Vantage applications as viewers in the same exercise, except for standard HLA compatibility issues. However, a VR-Vantage application can only control display engines from the same release. For example, a VR-Vantage 3.0.3 application could not control a VR-Vantage 2.8 display engine. However, applications and display engines built with different Windows compilers can interoperate.

1.6.6 MAK Data Logger

Since the Data Logger only receives data from the network or sends data over it, there are no compatibility issues for coexisting on the network with other versions. Recordings made with any version of the Logger in the 3.x series can be played on any later version of the Logger. Recordings made with the HLA versions of Logger 3.9 or later cannot be played on versions 3.8.2-ngc or earlier. Recordings made with Logger 4.x or later cannot be played on a 3.x Logger. HLA recordings made using version 3.7 and earlier of the Logger (and possibly some later versions) cannot be played on Logger 4.x or later. If you open these older recordings in Logger 3.12 and save them, they will work in Logger 4.x or later. The Logger file format changed in Logger 5.3 and 5.4. If you open an older file, you are prompted to update to the new format, which is handled automatically.

Recordings are protocol specific. For example, the HLA Logger cannot play a recording made with the DIS Logger. If you need to use a recording made with one protocol with an application from the other, use the MAK Gateway to translate the recording's output to the other protocol.

Use of HLA Time Management during recording does not affect the actual recording process. A recording made with Time Management enabled can be played back with or without Time Management. Similarly, a recording made without Time Management can be played back with or without Time Management.

If you are using HLA Evolved, see the restrictions in the note under "1.3. Running HLA Applications" on page 10.

Recording HLA 1.3 Logger Files and Playing Back in HLA 1516 Federations

In general, the MAK Data Logger uses the same file format for HLA 1.3 and HLA 1516 Logger files. You can record a Logger file from an HLA 1.3 federation, and play it back into an HLA 1516 federation (and vice versa). However, you must use the same FOM representation during record and playback. For example, if you use an HLA 1.3-style FED file during recording, you must use an HLA 1.3-style FED file during playback, even if you are using the IEEE 1516 version of the Logger and an IEEE 1516 RTI. (VR-Link and the MAK RTI allow you to use HLA 1.3-style FED files with IEEE 1516-based federates, and vice versa).

The main reason that consistency in FOM format is necessary is that IEEE 1516 uses different names for the "Root" classes of the Object and Interaction class hierarchies. A 1.3-style FED file requires a Root class called Object-Root, whereas a 1516-style XML files requires a Root class called HLAObjectRoot. The reason this can be a problem is that if the Logger is playing an HLA 1.3-based Logger file into a federation that is using a 1516-based XML file, it might come across an instance of a class called, for example,

"ObjectRoot.Vehicle". If it tries to register an object of this class, the RTI will complain that no such class exists. There might be a class called HLAObjectRoot.Vehicle in the current FOM, but the RTI or Logger does not know that this is actually the same class.

In future versions of the Logger, we intend to rectify this situation and allow playback of FED-based Logger files in federations that use XML-based FOM representations (as long as the FOMs otherwise describe a consistent set of classes, attributes, and parameters).

1.6.7 VR-Exchange

Since VR-Exchange is only an executable, it has no version compatibility issues. Also, it should be less susceptible to library and compiler changes on Linux than other MAK ONE products. On Windows, it has the same compiler compatibility issues as described in "1.5.2 Development Environment Compatibility" on page 12. If you use a FOM Mapper with VR-Exchange, it must be built with the same version of VR-Link as the version of VR-Exchange that you are running it with.

If you are using HLA Evolved, see the restrictions in the note in "1.3. Running HLA Applications" on page 10.

1.6.8 WebLVC Server

WebLVC Server translates simulation data between the WebLVC Protocol and traditional modeling and simulation protocols such as DIS and HLA. It is based on MAK's VR-Exchange interoperability platform, which supports many-to-many translation between various interoperability protocols through the use of "Brokers" for each protocol.

WebLVC contains optional plug-ins that are compatible only with specific versions of VR-Forces. See the product documentation or [Product Specifications & Dependencies](#).

1.6.9 VR-TheWorld

VR-TheWorld has no dependencies on other MAK ONE products. It does not use DIS or HLA. The URL for VR-Vantage users on the web interface requires VR-Vantage 1.2 or later. However, VR-Vantage 1.1, which supports terrain servers, can connect to VR-TheWorld.

1.7. Inter-Product Compatibility

This section discusses compatibility between different MAK ONE products.

In general, to build an application using one of our APIs, you must link with the appropriate version of VR-Link, and (for HLA applications) an RTI. You must be using a supported operating system with the correct compiler. These requirements are listed in the release notes for each product and on the Product Specifications & Dependencies page on the MAK website: <https://www.mak.com/mak-one-specs>.

On Windows, the correct versions of VR-Link libraries (.dll and .lib) are shipped with each MAK ONE product.

DIS versions of the Logger, VR-Vantage, and VR-Link applications should all be able to interact with each other and with other MAK ONE products without any difficulty. HLA versions should also be able to interact without problems assuming that the HLA environment is set up properly (same RTI version, same FED file.)

VR-Forces should interoperate with different DIS versions of the Logger, VR-Vantage, and VR-Link applications without difficulty. It should also interoperate with HLA versions, subject to the issues previously discussed.

Stealth control from VR-Vantage and VR-Forces uses Stealth Control PDUs. They should work with any version of the Stealth that supports them.

 Stealth version 6.2 and earlier supports the full list of Stealth Control PDUs. Stealth 7.0 (part of VR-Vantage) and VR-Vantage Stealth support a limited set of these PDUs. The Stealth Control Toolkit has been superseded by the VR-Vantage Control Toolkit. Please see the *VR-Vantage User Guide* for details.

Windows versions built with different versions of the C++ compiler may have problems running together on the same computer.

1.7.1 VR-Engage

A VR-Engage release is built against a specific version of VR-Forces. If you want a release of VR-Engage to interact with VR-Forces, that is to take control of entities being simulated by VR-Forces, they must be compatible releases. They must have the same SMSs and the VR-Engage add-ons for VR-Forces must be installed in VR-Forces. See the *VR-Engage User Guide* documentation to learn which version of VR-Forces is compatible with your version of VR-Engage.

1.7.2 Recording VR-Forces Simulations with the MAK Data Logger

You can use the MAK Data Logger to record VR-Forces simulations and then play them back for review. If you want to view a Logger file in VR-Forces (as opposed to some other MAK viewer, such as VR-Vantage), note these considerations:

- When you view the Logger file, you must run VR-Forces with an application number that is different from the one it used when you recorded the simulation. If you use the same application number, VR-Forces may crash. (We recommend that when you record the simulation, you run VR-Forces with an application number other than

the default (or whatever application number you typically use). This reduces the likelihood of a conflict when you play back the Logger file. You will not have to remember to change the VR-Forces application number every time you start it.)

- Do not load the original scenario in VR-Forces while you view the recording. Just load the terrain database that the simulation requires.
- If you are using HLA 1516, do not record a simulation and then immediately play it back. Exit the applications and start a new federation. Trying to play back the file in the federation from which it was recorded can result in name reservation conflicts and applications may crash.

1.7.3 FED File Naming Conventions

Prior to version 5.2, VR-Link named FED files using the convention *VR-Link<RPR FOM version>.fed* or *VR-Link<RPR FOM version>.xml*. For example, the FED file for RPR FOM 2, draft 17 was *VR-Link20017-1.fed*. This naming scheme had two problems. First, it was confusing for end-users of applications that use VR-Link (but who are not themselves VR-Link customers) to be using a FOM named that way. Second, the filename was not versioned, so any updates that we would make to the existing FOMs could cause compatibility problems. In fact, for this latter reason, we had refrained from fixing a few minor bugs that we found over the years until now.

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

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

VR-Link continues to include FED files and XML files using the old convention, so you do not have to change any existing scripts or code. At some point, the old FED files will be deprecated.

The default startup configurations for MAK ONE products built against VR-Link 5.2 or later use the new naming conventions. Therefore, if you want a newer release of a product to interoperate with one using the old naming conventions, you must change the connection configurations so that your applications all use the same FED files and federation names.

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

Table 1. Old FED file to new FED file mappings

Old FOM	New FOM
HLA 1.3	
DI-Guy-13_0.fed	

Table 1. Old FED file to new FED file mappings (continued)

Old FOM	New FOM
DI-Guy.fed	
VR-Link.fed	MAK-RPR1-1-1.fed
VR-Link05.fed	
VR-Link07.fed	
VR-Link08.fed	
VR-Link10.fed	MAK-RPR1-1-1.fed
VR-Link16.fed	
VR-Link20006.fed	
VR-Link20014.fed	
VR-Link20017-1.fed	MAK-RPR20017-1-1.fed
VR-Link20017-6.fed	MAK-RPR20017-6-1.fed
VR-Link20017.fed	
VR-Link20020.fed	MAK-RPR2-1-1.fed
VR-LinkBOML16.fed	
VR-LinkDDM.fed	
VR-LinkMC02.fed	MAK-RPR1-MC02-1-1.fed
VR-LinkTM.fed	MAK-RPR1-TimeManagement-1-1.fed
VRL-20017-1-Link16.fed	
VRL-DIGuy20017.fed	
VRL-DIGuy20017DDM.fed	
1516-2000	
DI-Guy-13_0.xml	
DI-Guy.xml	
VR-Link.xml	MAK-RPR1-1-1.xml
VR-Link20014.xml	
VR-Link20017-1.xml	MAK-RPR20017-1-1.xml
VR-Link20017-6.xml	MAK-RPR20017-6-1.xml
VR-Link20020.xml	MAK-RPR2-1-1.xml
VR-LinkBOML16.xml	
VR-LinkDDM.xml	
VR-LinkTM.xml	MAK-RPR1-TimeManagement-1-1.xml
VRL-DIGuy20017.xml	

Table 1. Old FED file to new FED file mappings (continued)

Old FOM	New FOM
VRL-DIGuy20017DDM.xml	
HLA Evolved	
DIGuyModule_1300.xml	MAK-DIGuy-1_evolved.xml
DIGuyModule_1302.xml	
RPR_FOM_v2.0_draft20_1516-2010.xml	RPR_FOM_v2.0_1516-2010.xml
VR-Link20014_base_evolved.xml	
VR-Link20014_evolved.xml	
VR-Link20017-1_base_evolved.xml	MAK-RPR20017-Base-1-1_evolved.xml
VR-Link20017-1_base_evolved_withUpdateRate.xml	
VR-Link20017-1_evolved.xml	MAK-RPR20017-1-1_evolved.xml
VR-Link20017-1_evolved_withUpdateRate.xml	
VR-Link20017-6_base_evolved.xml	MAK-RPR20017-Base-6-1_evolved.xml
VR-Link20017-6_evolved.xml	MAK-RPR20017-6-1_evolved.xml
VR-LinkDDM_base_evolved.xml	
VR-LinkDDM_evolved.xml	
VR-Link_LgrControl_evolved.xml	MAK-LgrControl-1_evolved.xml
VR-Link_ViewControl_evolved.xml	
VR-Link_VrlExt_evolved.xml	MAK-VRFExt-1_evolved.xml
VR-Link_base_evolved.xml	MAK-RPR1-Base-1-1_evolved.xml
VR-Link_evolved.xml	MAK-RPR1-1-1_evolved.xml
VRL-DIGuy20017DDM_evolved.xml	
VRL-DIGuy20017_evolved.xml	
updateRate.xml	MAK-updateRate-1_evolved.xml
VR-LinkBOML16_module.xml	MAK-BOML16-1_evolved.xml
	MAK-VRFAggregate-1_evolved.xml

1.7.4 Compatibility Tables

To see the compatibility table that cross-references all MAK ONE products, showing which ones are compatible, please visit the Build Compatibility page on the MAK Technologies website: <https://www.mak.com/build-compatibility>.

1.8. The Exercise Connection Configuration File

The exercise connection configuration (exConnConfig) file, *MAK-ONE-YYYY-Config.xml*, specifies the basic set of connection settings for a set of MAK ONE product releases. You can create multiple configuration files and use them for different exercises or environments, and then simply specify that the MAK ONE product use the desired configuration file.

 If you share the same file between multiple MAK ONE applications and include application-specific settings in the file, there may be undesired results.

1.8.1 Exercise Configuration File Parameters

The file is in the product installation directory. For VR-Link, it is in *./bin64*, with a backup copy in *./data/config*. For other applications, it is in *./appData/settings/connections*, with a backup in *./factory/settings/connections*.

The HLA connection parameters are:

- Federation Name. Specifies the name of the federation execution.
- FED File Name. Specifies the FED file to use.
- RPR FOM Version. Specifies the RPR FOM version.
- RPR FOM Revision. Specifies the RPR FOM revision.
- NETN FOM Version. Specifies the NETN FOM version as a double. The default (and currently the only supported version) is 3.0.
- NETN FOM Revision. Specifies the internal revision of the implementation of the NETN FOM version as an integer. The default (and currently the only supported revision) is 1.
- FOM Modules. Specifies one or more HLA Evolved or HLA 4 FOM modules. It can take multiple entries, which you should list in their hierarchical order to prevent any issues related to dependencies.

The DIS connection parameters are:

- Exercise ID. Specifies the DIS exercise ID.
- DIS Version. Specifies the DIS version. 7 is currently the only valid setting.
- Use IPV6. Specifies that the DIS exercise connection use the IPV6 protocol, if available.

1.8.2 Specifying Exercise Connection Configuration Files on the Command Line

You can specify the exercise configuration file to use when launching a MAK ONE application with the `--exConnConfigFile filename` command-line option. If you do not specify a file, the application uses the *MAK-ONE-YYYY-Config.xml* file.

If you have a file for your needs, you should specify that file on the command line. The specified configuration file takes precedence over the *MAK-ONE-YYYY-Config.xml* file, replacing default values. It is critical to include all necessary parameters in the exercise connection configuration file to ensure interoperability between simulation applications. For example, on the command line:

```
vrfsimprotocol --exConnConfigFile myConfig.xml
```

where:

- *protocol* is DIS, HLA1516e, or HLA4.
- *YYYY* is the year in the configuration file included in your installation.
- *myConfig.xml* is your custom configuration file.

 The *MAK-ONE-YYYY-Config.xml* file is always loaded to ensure that no parameters are missing.

1.9. Synchronizing Terrain for the VR-Vantage and VR-Forces

Many MAK Technologies customers create scenarios in VR-Forces and view them in one of the VR-Vantage applications. The terrain databases used by the two applications must be synchronized so that the 3D view matches the 2D view and the simulation engine.

VR-Forces 4.0 and later use the same terrain libraries as VR-Vantage. Therefore, if the VR-Forces GUI and sim engine use the same terrain as VR-Vantage, simulation and visualization are fully synchronized. However, there may be occasions in which you use different terrains in the GUI and sim engine, in which case the synchronization issues discussed in this section would still apply.

Terrain problems may occur under the circumstances described in the following sections.

1.9.1 Flat Earth Databases

When VR-Forces 3.x imports a Flat Earth database, it converts it to UTM; VR-Vantage loads it as-is. If the database is larger than one UTM zone, discrepancies between the VR-Vantage view and VR-Forces will occur in areas outside the UTM zone.

1.9.2 Unsupported Features

VR-Vantage supports virtually all FLT features. VR-Forces 3.x does not support all FLT features. If a database has unsupported features, VR-Forces may not be able to load it.

1.9.3 Database is not WGS84

VR-Vantage assumes that all databases use WGS84 datum. It does not check the database header. VR-Forces 3.x checks the database header and responds appropriately. If you are using a database that does not use WGS84 datum, the best option is to convert it to WGS84. If you cannot convert the database, setting it to WGS84 will provide correlation between VR-Vantage and VR-Forces, however the coordinates sent over the network will be incorrect. For assistance, please contact support at <https://www.mak.com/mak-one/support/portal>.

 Any time you use non WGS84 data with VR-Forces in DIS, the coordinates will be incorrect on the network because DIS expects WGS84 data.

1.9.4 Optimized Databases

Databases can be optimized for 3D use or for simulation. If a database is optimized for one, it may work less well in the other. For example, a large, complex database could be optimized for 3D viewing through the use of several levels of detail. While it might work very well in VR-Vantage, it could cause a problem for VR-Forces 3.x performance due to its size.

1.10. Delivering Applications to Users

A developer's license (for example, VR-Link or VR-Forces) allows you to compile applications using the product libraries and run one such application. After you have developed an application, you may need to deliver that application to an internal or external customer. The end user of the application needs a runtime license to use it. The customer does not need the entire toolkit, nor do they need a developer's license (unless they need to recompile the application.) They need only the application, the product libraries (if you link dynamically), the license management files, and a runtime license for each instance of the application that they will run. You should not provide the header files. If you statically link to the MAK libraries, you do not have to deliver them, either.

You can purchase a runtime license for your customer or have the customer purchase the license from MAK, depending on the arrangements you have made.



2. The MAK Object Model

The object model used in MAK ONE products is based on the NATO Education and Training Network (NETN) Federation Object Model (FOM) 3.0 (<https://github.com/AMSP-04/NETN-FOM>), which in turn is based on the Real-time Platform Reference (RPR) FOM 2.0, and that is based on the Distributed Interactive Simulation (DIS) standard.

2.1. MAK Object Model Overview	24
2.2. DIS and RPR FOM Support	25
2.3. NETN FOM Support	27
2.4. MAK Object Model Extensions	29
2.4.1 Aerodromes and Runways	29
2.4.2 Aggregates	33
2.4.3 Dynamic Terrain	33
2.4.4 Human Characters (DI-Guy)	38
2.4.5 IFF	39
2.4.6 Logger Control	39
2.4.7 Physical Entities	39
2.4.8 Radar	46
2.4.9 Sensor Field of View	47
2.4.10 Time and Date	49
2.4.11 Weather	51

2.1. MAK Object Model Overview

MAK ONE products use an object model based on the NETN FOM, but they do not support all functionality in those object models. The VR-Link API supports most of the classes in DIS and the RPR FOM, but other products such as VR-Forces and VR-Vantage only support a subset. For example, VR-Forces does not support the DIS/RPR FOM minefield implementation. There are also cases, such as the Start/Resume interaction, where the capability is partially, but not fully, supported.

MAK has also added many extensions on top of these standard object models. These cover new concepts or additional attributes that are not present in the base object models. This chapter provides information on these extensions

In DIS, MAK ONE products support the object model defined in DIS 7. Extensions to DIS are added in two ways: via custom PDUs and custom records. See the sections in this chapter for details.

In HLA Evolved and HLA 4, MAK ONE products support the NETN FOM plus MAK extensions. Extensions to the standard FOMs are added using custom FOM classes as well as custom records. These extensions are separated into different FOM modules, based on content. The names of all MAK-developed modules begin with the prefix "MAK-". They also include the version number in their name, for example, the "-2" in *MAK-Physical-2_evolved.xml*. Any incompatible changes to a FOM module that are made between releases will cause this version number to be incremented.

 HLA 4 versions of MAK ONE products use the HLA Evolved FOM files for interoperability purposes. Future HLA 4 versions of MAK ONE products will have HLA 4-specific FOM files.

MAK ONE products use these FOM modules:

- RPR_FOM_v2.0_1516-2010.xml
- NETN-BASE.xml
- NETN-Physical.xml
- NETN-METOC.xml
- NETN-MRM.xml
- MAK-Physical-2_evolved.xml
- MAK-Aerodrome-1_evolved.xml
- MAK-METOC-1_evolved.xml
- MAK-VRFExt-9_evolved.xml
- MAK-DIGuy-7_evolved.xml
- MAK-LgrControl-2_evolved.xml
- MAK-VRFAggregate-6_evolved.xml
- MAK-DynamicTerrain-2_evolved.xml

- MAK-VRLExt-3_evolved.xml
- MAK-DER-1_evolved.xml
- RPR-Enumerations_Experimental_IFF.xml
- RPR-MAK_Experimental_IFF-3.xml

2.2. DIS and RPR FOM Support

The following DIS 7 PDUs and RPR FOM 2.0 classes are supported in VR-Forces.

The unsupported PDU types and RPR classes are supported by the VR-Link API and can be added to VR-Forces through the API toolkit.

Most PDUs and FOM classes have many fields/attributes, multiple aspects, and potentially several use cases. For those listed as having support in VR-Forces, VR-Forces does not necessarily react to, send, or support all these fields, attributes, aspects, and use cases.

Table 2. Supported DIS PDUs and RPR FOM classes

DIS PDU	PDU Type	RPR FOM Class	VR-Forces Support
Entity State	1	PhysicalEntity subclasses	Send / Receive
Collision	4	Collision	None
Collision-Elastic	66	CollisionElastic	None
Entity State Update	67	-	Receive
Attribute	72	-	Send / Receive
Fire	2	WeaponFire	Send / Receive
Detonation	3	MunitionDetonation	Send / Receive
Directed Energy Fire	68	-	None
Entity Damage Status	69	-	None
Service Request	5	ServiceRequest	None
Resupply Offer	6	ResupplyOffer	Send / Receive
Resupply Received	7	ResupplyReceived	Send / Receive
Resupply Cancel	8	ResupplyCancel	Send / Receive
Repair Complete	9	RepairComplete	None
Repair Response	10	RepairResponse	None
Create Entity	11	CreateEntity	None

Table 2. Supported DIS PDUs and RPR FOM classes (continued)

DIS PDU	PDU Type	RPR FOM Class	VR-Forces Support
Remove Entity	12	RemoveEntity	None
Start/Resume	13	StartResume	Send / Receive
Stop/Freeze	14	StopFreeze	Send / Receive
Acknowledge	15	Acknowledge	None
Action Request	16	ActionRequest	None
Action Response	17	ActionResponse	None
Data Query	18	DataQuery	None
Set Data	19	SetData	None
Data	20	Data	Send / Receive
Event Report	21	EventReport	None
Comment	22	Comment	Send
Electromagnetic Emission	23	EmitterSystem / EmitterBeam	Send / Receive
Designator	24	Designator	Send / Receive
Underwater Acoustic (UA)	29	ActiveSonar / PropulsionNoise/ ActiveSonarBeam	Send / Receive
IFF	28	IFFSystem subclasses / IFFData subclasses	Send / Receive
<i>i</i> The IFFSystem and IFFData subclasses are RPR FOM 3.0 classes that have been ported down to RPR FOM 2.0.			
Supplemental Emission/Entity State (SEES)	30	PhysicalEntity	None
Transmitter	25	RadioTransmitter	Send / Receive
Signal	26	RadioSignal	Send / Receive
Receiver	27	RadioReceiver	Send
Intercom Signal	31	-	None
Intercom Control	32	-	None
Aggregate State	33	AggregateEntity	Send / Receive

Table 2. Supported DIS PDUs and RPR FOM classes (continued)

DIS PDU	PDU Type	RPR FOM Class	VR-Forces Support
IsGroupOf	34	-	None
Transfer Ownership	35	TransferOwnership	None
IsPartOf	36	PhysicalEntity	None
Minefield State	37	Minefield	None
Minefield Query	38	MinefieldQuery	None
Minefield Data	39	MinefieldData	None
Minefield Response Negative Acknowledgment (NACK)	40	MinefieldResponseNack	None
Environmental Process	41	EnvironmentProcess	Send / Receive
Gridded Data	42	GriddedData	None
Point Object State	43	PointObject	None
Linear Object State	44	LinearObject	None
Areal Object State	45	ArealObject	None
Create Entity-R	51	CreateEntityR	None
Remove Entity-R	52	RemoveEntityR	None
Start/Resume-R	53	StartResumeR	None
Stop/Freeze-R	54	StopFreezeR	None
Acknowledge-R	55	AcknowledgeR	None
Action Request-R	56	ActionRequestR	None
Action Response-R	57	ActionResponseR	None
Data Query-R	58	DataQueryR	None
Set Data-R	59	SetDataR	None
Data-R	60	DataR	None
Event Report-R	61	EventReportR	None
Comment-R	62	CommentR	None
Record Query-R	65	RecordQueryR	None
Set Record-R	64	SetRecordR	None
Record-R	63	RecordR	None

2.3. NETN FOM Support

The NATO Education and Training FOM extends the RPR FOM with many new FOM modules that contain a wide array of new classes and subclasses. VR-Forces uses the NETN FOM by default, but it only uses a small number of these classes and attributes. For Physical and MRM object classes, VR-Forces supports only the Callsign and UniqueId attributes. For limitations on METOC and ETR support, please see "Differences Between

NETN METOC and MAK METOC Object Models" on page 51 and "NETN ETR Tasks" in the *VR-Forces Developers Guide* (under Tasks and Sets in the Simulation Object Design section).

Supported classes:

- NETN_Aircraft (Physical)
- NETN_AmphibiousVehicle (Physical)
- NETN_GroundVehicle (Physical)
- NETN_MultiDomainPlatform (Physical)
- NETN_Spacecraft (Physical)
- NETN_SubmersibleVessel (Physical)
- NETN_SurfaceVessel (Physical)
- NETN_Human (Physical)
- NETN_NonHuman (Physical)
- NETN_CulturalFeature (Physical)
- NETN_Munition (Physical)
- NETN_Aggregate (MRM)
- LandSurface (METOC)
- WaterSurface (METOC)
- TroposphereLayer (METOC)
- SubsurfaceLayer (METOC)
- Service (METOC)
- MagicMove (ETR)
- MagicResource (ETR)
- MoveToLocation (ETR)
- FireAtEntity (ETR)
- FireAtLocation (ETR)
- Mount (ETR)
- Dismount (ETR)
- SetOrderedAltitude (ETR)
- SetOrderedSpeed (ETR)
- SetRulesOfEngagement (ETR)

2.4. MAK Object Model Extensions

MAK uses the following extensions to the standard object models.

2.4.1 Aerodromes and Runways

The Aerodrome extensions are designed to transmit information about Aerodrome objects (sometimes simply referred to as airports). An Aerodrome object can be associated with one or more Runway objects to describe the state of its runways. The attributes included in the Aerodrome object describe the light state specific to the aerodrome.

Runway objects can exist on their own, but are usually associated with Aerodrome objects. Runway objects include attributes to describe the light state and condition of the runway.

DIS

The Aerodrome object model extensions are represented by the following custom PDUs.

Aerodrome PDU

PDU Kind: 222

Table 3. Aerodrome PDU Contents

Field Size (bits)	MAK Aerodrome PDU Fields	
96	PDU Header	Protocol Version. 8-bit enumeration
		Exercise ID. 8-bit unsigned integer
		PDU Type = 222. 8-bit enumeration
		Protocol Family. 8-bit enumeration
		Timestamp. 32-bit unsigned integer
		Length. 16-bit unsigned integer
		PDU Status. 8-bit record
		Padding. 8 bits, unused
16	Version	Final PDU Status. Most significant bit
		MAK INTERNAL. 15-bit unsigned integer
48	Entity Identifier	Site Number. 16-bit unsigned integer
		Application Number. 16-bit unsigned integer
		Entity Number. 16-bit unsigned integer
128	Unique Identifier	Universally Unique Identifier. 128 bits
8	Force Identifier	8-bit unsigned integer

Table 3. Aerodrome PDU Contents (continued)

Field Size (bits)	MAK Aerodrome PDU Fields	
8	Active	8-bit boolean
8	Apron Light	8-bit unsigned integer
8	Apron Flood Light	8-bit boolean
8	Clearance Bar Light	8-bit unsigned integer
8	Taxiway Center Light	8-bit unsigned integer
8	Taxiway Edge Light	8-bit unsigned integer
8	Taxiway Identifier Marker Board Light	8-bit unsigned integer
8	Undercarriage Inspection Light	8-bit boolean
32	Airport Code	8-bit character
		8-bit character
		8-bit character
		8-bit character

Runway PDU

PDU Kind: 223

Table 4. Runway PDU Contents

Field Size (bits)	MAK Runway PDU Fields	
96	PDU Header	Protocol Version. 8-bit enumeration
		Exercise ID. 8-bit unsigned integer
		PDU Type = 223. 8-bit enumeration
		Protocol Family. 8-bit enumeration
		Timestamp. 32-bit unsigned integer
		Length. 16-bit unsigned integer
		PDU Status. 8-bit record
		Padding. 8 bits, unused
16	Version	Final PDU Status. Most significant bit MAK INTERNAL. 15-bit unsigned integer
48	Entity Identifier	Site Number. 16-bit unsigned integer
		Application Number. 16-bit unsigned integer
		Entity Number. 16-bit unsigned integer

Table 4. Runway PDU Contents (continued)

Field Size (bits)	MAK Runway PDU Fields	
128	Unique Identifier	Universally Unique Identifier. 128 bits
128	Aerodrome Unique Identifier	Universally Unique Identifier. 128 bits
8	Damaged	8-bit boolean
8	Condition	8-bit enumeration
8	Runway 1 Active	8-bit boolean
8	Runway 2 Active	8-bit boolean
192	Endpoint 1	X-component. 64-bit floating point
		Y-component. 64-bit floating point
		Z-component. 64-bit floating point
192	Endpoint 2	X-component. 64-bit floating point
		Y-component. 64-bit floating point
		Z-component. 64-bit floating point
24	Runway 1 Name	8-bit character
		8-bit character
		8-bit character
24	Runway 2 Name	8-bit character
		8-bit character
		8-bit character
8	Runway 1 Center Line Light	8-bit unsigned integer
8	Runway 2 Center Line Light	8-bit unsigned integer
8	Runway 1 Distance To Go Light	8-bit unsigned integer
8	Runway 2 Distance To Go Light	8-bit unsigned integer
8	Runway 1 Edge Light	8-bit unsigned integer
8	Runway 2 Edge Light	8-bit unsigned integer
8	Runway 1 Displaced Threshold Light	8-bit unsigned integer
8	Runway 2 Displaced Threshold Light	8-bit unsigned integer
8	Runway 1 End Light	8-bit unsigned integer
8	Runway 2 End Light	8-bit unsigned integer
8	Runway 1 Touchdown Zone Light	8-bit unsigned integer
8	Runway 2 Touchdown Zone Light	8-bit unsigned integer

Table 4. Runway PDU Contents (continued)

Field Size (bits)	MAK Runway PDU Fields	
8	Runway 1 Threshold Lighting Light	8-bit unsigned integer
8	Runway 2 Threshold Lighting Light	8-bit unsigned integer
8	Runway 1 End Indicator Light	8-bit unsigned integer
8	Runway 2 End Indicator Light	8-bit unsigned integer
8	Runway 1 Approach Light	8-bit unsigned integer
8	Runway 2 Approach Light	8-bit unsigned integer
8	Runway 1 VGSI Light	8-bit unsigned integer
8	Runway 2 VGSI Light	8-bit unsigned integer
8	Runway 1 Rapid Exit Taxiway Light	8-bit unsigned integer
8	Runway 2 Rapid Exit Taxiway Light	8-bit unsigned integer
8	Runway 1 Runway Exit Light	8-bit unsigned integer
8	Runway 2 Runway Exit Light	8-bit unsigned integer
8	Runway 1 Taxiway Lead On Light	8-bit unsigned integer
8	Runway 2 Taxiway Lead On Light	8-bit unsigned integer
8	Runway 1 Taxiway Stop Bar Light	8-bit unsigned integer
8	Runway 2 Taxiway Stop Bar Light	8-bit unsigned integer
8	Runway 1 Prethreshold Edge Light	8-bit unsigned integer
8	Runway 2 Prethreshold Edge Light	8-bit unsigned integer
8	Runway 1 Arrestor Gear Up	8-bit unsigned integer
8	Runway 2 Arrestor Gear Up	8-bit unsigned integer
8	Padding	8-bit unsigned integer

HLA

Aerodromes and Runways are represented by custom object classes in HLA. For full details on the classes and their attributes, please see the *MAK-Aerodrome-1_evolved.xml* FOM module.

2.4.2 Aggregates

The Aggregate object model extensions were created to support the aggregate-level modeling in VR-Forces. They add higher-level concepts such as personnel, equipment, and supplies, as well as interactions for units to affect one another.

DIS

Because the VR-Forces aggregate-level model is not supported in DIS, there are no defined PDUs or records to transmit this data.

HLA

The aggregate-level modeling extensions define a new subclass for the AggregateEntity class, as well as new object classes to represent things such as engineering objects and air bases (not that the model used in aggregate-level modeling is different than the aerodrome model used in entity-level modeling). There are also a set of Influence interactions, which are used to represent the ways one aggregate unit affects another. This includes interactions such as engagements, construction of engineering objects, jamming, and resupply. For full details on the classes and their attributes, please see the *MAK-VRFAggregate-6_evolved.xml* FOM module.

2.4.3 Dynamic Terrain

The Dynamic Terrain extensions provide a way for an application such as VR-Forces to simulate changes to the terrain and transmit them on the network for other applications to deform the terrain in the same manner. The extension is flexible enough to support a variety of types of changes.

The easiest way to use Dynamic Terrain is to use the VR-Forces remote control toolkit, which provides developers with an API to send and receive this data. If you are interested in learning more, read the extension descriptions for DIS and HLA that follow.

Each published dynamic terrain object includes one or more terrain geometries that define the areas of the terrain that are being changed. For each terrain geometry, a list of key/value pairs are specified. These key/value pairs are ASCII strings that define the type of change and the new state. This could, for example, identify a new state for switch nodes in that area of the terrain, or tell VR-Forces how to programmatically change the terrain. The exact values which these keys and values use will depend on the terrain and how VR-Forces is configured, but the most commonly used ones are described in the section "Geometry Data" on the next page.

Dynamic Terrain Object Attributes and Fields

While the network representation is different for HLA and DIS, the primary attributes of the Dynamic Terrain object are the same in both protocols. All of the following attributes apply to both protocols, except where noted.

To understand how the attributes are related, see Table 6 on page 36.

PDU Version (DIS only)

This field indicates the version of the PDU. When VR-Forces makes changes to the PDU between releases, this number is increased so that applications can understand how to properly decode the data contained in the PDU. This document describes the format of

PDU version 1.

Final Update (DIS only)

This is a boolean value that indicates whether this is the final update for this dynamic terrain object. It indicates that the dynamic terrain changes defined by this object should be immediately removed. In HLA, this processing should happen when the object is deleted.

Geometry and Data

This is a list of geometry records and associated terrain changes. It describes where the terrain is being changed and how it is being changed. The list can contain a single terrain change, or multiple.

The first subfield in the Geometry and Data is a 32-bit integer specifying the number of entries in the list. Following this is the list of each Geometry and Data Entry in order.

Geometry and Data Entry

A Geometry and Data Entry defines a single terrain geometry and some associated terrain changes. It consists the subfields described below:

- Geometry
- Geometry Data
- Event Number
- Priority

Once a Geometry and Data Entry has been added, it will never be removed because it can be uniquely identified by its index in the Geometry and Data list. When a Geometry and Data Entry should be removed, the Geometry Data will be cleared.

Geometry

Defines the terrain geometry of a change. The first field, Terrain Area Type, is a 32-bit integer enumeration that specifies how the geometry is defined, for example, a circle, a polygonal area, or line with width. Currently only circular geometries are supported, which is a value of 1.

For circular geometries, this is followed by a location, expressed in geocentric coordinates, and a radius, expressed in meters.

Geometry Data

Defines how the terrain at the specified geometry should be changed. This is a list of changes. The first field is a 32-bit integer, specifying the number of changes. Following that is a list of key/value pairs, where each key and value is an ASCII string. The first 32 bits of each string is an integer that specifies the number of characters in the string. Following that are the characters of the string.

The specific keys and values to be used will depend on your terrain and what switches it expects. Different terrains support different types of dynamic changes. For example, a specific key/value pair may be "damage"/"slight". This indicates that any dynamic terrain features within the specified geometry should have their "damage" state changed to "slight". The table below lists the common types of dynamic terrain changes supported by MAK ONE products in a default configuration, but new types of changes can be added through configuration and use of new terrains.

Table 5. Dynamic Terrain Key/Value Pairs

Key	Allowable Values	Description
damage	none, slight, moderate, destroyed	General terrain damage.
damage_structure	none, slight, moderate, destroyed	Damage specific to buildings and structures.
door	open, closed	The state of a door.
window	open, closed	The state of a window.

Event Number

This field starts with a value of 0 the first time this particular Geometry and Data Entry updates. Each subsequent time it updates, the value increments by 1. This allows an application to monitor a single field to determine whether any update has been made to this particular dynamic terrain change. This saves the application the work of comparing its stored state with every field in the Geometry and Data Entry.

Priority

Specifies the priority of this terrain change relative to other terrain changes. If two different Geometry and Data Entries conflict with one another, the one with the higher priority takes precedence.

Common Data

Common Data is similar to the Geometry Data, described in "Geometry Data" on the previous page. It has the same format and contains a list of key/value pairs that define changes to the terrain. The difference is that Common Data applies to all Geometry and Data entries described in this object. This can be useful if the same changes are being described for multiple different terrain geometries.

Common Data Event Number

This field starts with a value of 0 the first time Common Data is updated. Each subsequent time it updates, the value increments by 1. This allows an application to monitor a single field to determine whether any update has been made to the Common Data of this object. This saves the application the work of comparing its stored state with each key/value pair in the Common Data list.

DIS

VR-Forces sends a custom PDU that supports dynamic terrain changes. All updates are heartbeated periodically using normal DIS heartbeating rules.

Dynamic Terrain PDU

PDU Kind: 160

Table 6. Dynamic Terrain PDU Contents

Field Size (bits)	MAK Dynamic Terrain PDU Fields	
96	PDU Header	Protocol Version. 8-bit enumeration
		Exercise ID. 8-bit unsigned integer
		PDU Type = 160. 8-bit enumeration
		Protocol Family. 8-bit enumeration
		Timestamp. 32-bit unsigned integer
		Length. 16-bit unsigned integer
		PDU Status. 8-bit record
		Padding. 8 bits unused
48	Object Identifier	Site Number. 16-bit unsigned integer
		Application Number. 16-bit unsigned integer
		Entity Number. 16-bit unsigned integer
16	PDU Version	16-bit unsigned integer
32	Common Data Event Number	8-bit unsigned integer
8	Final Update	8-bit boolean
56	Padding	8-bit unsigned integer

Table 6. Dynamic Terrain PDU Contents (continued)

Field Size (bits)	MAK Dynamic Terrain PDU Fields			
variable	Geometry and Data	Length		32-bit unsigned integer
		Geometry And Data Entry 1	Geometry	32-bit unsigned integer
			Terrain Area Type	32 bits unused
			Padding	X-component. 64-bit floating point
			Location	Y-component. 64-bit floating point
				Z-component. 64-bit floating point
			Radius	64-bit floating point
			Geometry Data	Length
				32-bit unsigned integer
			Geometry Data Entry 1	Key string length. 32-bit unsigned integer
				Key string. Variable
				Value string length. 32-bit unsigned integer
				Value string. Variable
			...	
			Geometry Data Entry N	Same structure as Geometry Data Entry 1
		Event Number		32-bit unsigned integer
		Priority		32-bit unsigned integer
		Padding		Unused bits, padding to 64-bit boundary
		...		
		Geometry And Data Entry N		Same structure as Geometry and Data Entry 1

Table 6. Dynamic Terrain PDU Contents (continued)

Field Size (bits)	MAK Dynamic Terrain PDU Fields		
variable	Common Data	Length	32-bit unsigned integer
		Common Data Entry 1	Key string length. 32-bit unsigned integer
			Key string. Variable
			Value string length. 32-bit unsigned integer
			Value string. Variable
		...	
		Common Data Entry N	Same structure as Common Data Entry 1

HLA

Dynamic Terrain changes are represented by a custom object class in HLA, *DynamicTerrain*. For full details on the class and its attributes, please see the *MAK-DynamicTerrain-2_evolved.xml* FOM module.

2.4.4 Human Characters (DI-Guy)

Attributes of human characters are controlled by the DI-Guy object model extensions. These extensions provide information such as appearance, gestures, animations, and items held in the character's hands. This information is generally transmitted as strings. The expected string values depend on the DI-Guy character definitions. Use of these extensions is only supported through the VR-Link API.

DIS

All DI-Guy attributes are included as custom Attribute records in DIS. VR-Link uses these custom attribute record types:

- DtDiGuyActionRecord - 23000000
- DtDiGuyAppearanceRecord - 23000002
- DtDiGuyTypeRecord - 23000003
- DtDiGuyShapeVariationsRecord - 23000004
- DtDiGuyScaleRecord - 23000005
- DtDiGuyPatchesRecord - 23000006

Transmission and receipt of these records is only supported through use of the VR-Link API. Please see the *VR-Link Developer's Guide* for more details.

HLA

In HLA, the *MAK-DIGuy-7_evolved.xml* FOM module contains the DI-Guy extensions. The main class is the DIGuyObject object class. For every entity that represents a human character, VR-Forces will all create a DIGuyObject instance to represent the characteristics of this entity. This object references the entity object by EntityIdentifier and ObjectIdentifier.

Transmission and receipt of this class is only supported through use of the VR-Link API. Please see the *VR-Link Developer's Guide* for more details.

2.4.5 IFF

While DIS 7 supports IFF Mode 5 and Mode S, RPR FOM 2.0 does not. The RPR FOM 3.0 standard is adding support for these IFF modes, based on the object model in DIS 7, but with substantial changes to the structure of the IFF classes. The MAK FOM extensions have taken the RPR FOM 3.0 IFF classes and ported them for use with RPR FOM 2.0.

DIS

Support for IFF in DIS complies with the DIS 7 standard. No further extensions are required.

HLA

In HLA, the *RPR-MAK_Experimental_IFF-3.xml* and *RPR-Enumerations_Experimental_IFF.xml* FOM modules add the IFF classes and enumerations defined in the upcoming RPR FOM 3.0 standard for use with RPR FOM 2.0. For more information on these classes, please read the proposed changes for RPR FOM 3.0.

2.4.6 Logger Control

The MAK Data Logger is able to accept commands from remote applications using remote commands. In DIS, this is a custom PDU. In HLA, it is the LgrControl interaction class. Use of the Logger Control extensions is only supported through the logger control toolkit. Please see the VR-Link and MAK Data Logger documentation for more details on using this toolkit.

2.4.7 Physical Entities

MAK has added several additional attributes to the standard entity object model. Some of these attributes come from the NETN FOM, and thus are not present in standard DIS. Some are custom attributes used by MAK products. Some attributes are only used on certain types of entities (for example, aircraft), while some are sent for all entities.

In addition to the attributes described here, VR-Forces may send others as well. These include the VR-Forces internal attributes defined in the VrfExtendedAttributes object class, as well as state properties that can be transmitted in multiple ways. Please see the *VR-Forces Developers Guide* for more information on these.

DIS

Attributes added to DIS entities are all defined as new Attribute records. Please refer to the DIS standard and VR-Link documentation for details on using Attribute records.

DtUuidRecord

This Attribute record is based on the NETN FOM UniqueId attribute. It uniquely identifies an entity in the simulation and will be consistent on every run of an exercise. See the NETN FOM for more details.

Record Type: 24000000

Supported Entity Categories: All entities

Table 7. DtUuidRecord Contents

Field Size (bits)	UUID Record Fields	
32	Record Type (24000000)	32-bit enumeration
16	Record Length	16-bit unsigned integer
128	Unique Identifier	Universally Unique Identifier. 128 bits
8	Padding	8-bit unsigned integer

DtUuidSourceUnitRecord

This Attribute record is based on the NETN FOM SourceUnit attribute. See the NETN FOM for more details.

Record Type: 24000001

Supported Entity Categories: All entities

Table 8. DtUuidSourceUnitRecord Contents

Field Size (bits)	UUID Source Unit Record Fields	
32	Record Type (24000001)	32-bit enumeration
16	Record Length	16-bit unsigned integer
128	Unique Identifier	Universally Unique Identifier. 128 bits
8	Padding	8-bit unsigned integer

DtStatusRecord

This Attribute record is based on the NETN FOM Status attribute. See the NETN FOM for more details.

Record Type: 24000002

Supported Entity Categories: All entities

Table 9. DtStatusRecord Contents

Field Size (bits)	Status Record Fields	
32	Record Type (24000002)	32-bit enumeration
16	Record Length	16-bit unsigned integer
8	Status	8-bit enumeration
8	Padding	8-bit unsigned integer

DtActivityRecord

This Attribute record is based on the NETN FOM Activity attribute. See the NETN FOM for more details.

Record Type: 24000003

Supported Entity Categories: All entities

Table 10. DtActivityRecord Contents

Field Size (bits)	Activity Record Fields	
32	Record Type (24000003)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Activity	16-bit enumeration

DtEmbeddedUnitListRecord

This Attribute record is based on the NETN FOM EmbeddedUnitList attribute. See the NETN FOM for more details.

Record Type: 24000004

Supported Entity Categories: All entities

Table 11. DtEmbeddedUnitListRecord Contents

Field Size (bits)	Embedded Unit List Record Fields	
32	Record Type (24000004)	32-bit enumeration
16	Record Length	16-bit unsigned integer
128	Unique Identifier (U1)	Universally Unique Identifier. 128 bits
...		
128	Unique Identifier (UN)	Universally Unique Identifier. 128 bits
8	Padding	8-bit unsigned integer

N is the number of Unique Identifiers included in this record.

N is calculated by taking the Record Length value, removing the padding, type and length, and dividing by the size of a Unique Identifier

$N = (\text{RecordLength} - 5)/16$

DtDestinationRecord

This Attribute record is based on the NETN FOM Destination attribute. See the NETN FOM for more details.

Record Type: 24000005

Supported Entity Categories: All entities

Table 12. DtDestinationRecord Contents

Field Size (bits)	Destination Record Fields	
32	Record Type (24000005)	32-bit enumeration
16	Record Length	16-bit unsigned integer
8	Padding	8-bit unsigned integer
8	Padding	8-bit unsigned integer
32	Point Variant Type	32-bit enumeration
X	UUID or Geocentric Location	

The data for X depends on the value of the Point Variant Type. Point Variant Type has 2 supported values: 0 (Location), 1 (Unique Identifier)

Location is a Geocentric Location sent in the form of 3 doubles (x, y, z)

Unique Identifier is a 128-bit Universally Unique Identifier

DtRouteRecord

This Attribute record is based on the NETN FOM Route attribute. See the NETN FOM for more details.

Record Type: 24000006

Supported Entity Categories: All entities

Table 13. DtRouteRecord Contents

Field Size (bits)	Route Record Fields	
32	Record Type (24000006)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Padding	16-bit padding
32	Path Variant Type	32-bit enumeration
X	Path Variant Data	8-bit unsigned integer

Path Variant Data can have two defined types a Universally Unique Identifier and an Array of World Location Coordinates.

DtCallsignRecord

This Attribute record is based on the NETN FOM Callsign attribute. See the NETN FOM for more details.

Record Type: 24000007

Supported Entity Categories: All entities

Table 14. DtCallsignRecord Contents

Field Size (bits)	Callsign Record Fields	
32	Record Type (24000007)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Callsign Length	16-bit unsigned integer
8X	Callsign	Unicode String
P	Padding	8-bit unsigned integer
Callsign Length is in bytes. X is the Callsign Length. Padding must be inserted at end of the record to the nearest 64-bit boundary		

DtSymbolIdRecord

This Attribute record is based on the NETN FOM SymbolId attribute. See the NETN FOM for more details.

Record Type: 24000008

Supported Entity Categories: All entities

Table 15. DtSymbolIdRecord Contents

Field Size (bits)	Symbol Identifier Record Fields	
32	Record Type (24000008)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Symbol Identifier Length	16-bit unsigned integer
8X	Symbol Identifier	Unicode String
P	Padding	8-bit unsigned integer
Symbol Identifier Length is in bytes. X is the Symbol Identifier Length. Padding must be inserted at end of the record to the nearest 64-bit boundary		

DtAircraftLightingRecord

This Attribute record adds additional lighting for aircraft entities.

Record Type: 24000021

Supported Entity Categories: Aircraft

Table 16. DtAircraftLightingRecord Contents

Field Size (bits)	Aircraft Lighting Record Fields	
32	Record Type (24000021)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Version	MAK INTERNAL. 16-bit unsigned integer
8	Anticollision Lights DayNight	8-bit enumeration
8	Anticollision Lights Flashing	8-bit enumeration

Table 16. DtAircraftLightingRecord Contents (continued)

Field Size (bits)	Aircraft Lighting Record Fields	
8	Formation Lights Luminance	8-bit unsigned integer
8	Navigation Lights Flashing	8-bit enumeration
8	NVG Mode	8-bit enumeration
8	Taxiway Lights On	8-bit boolean
8	HDU Lights On	8-bit boolean
8	Padding	8-bit unsigned integer

DtTankerAircraftLightingRecord

This Attribute record adds additional lighting for tanker aircraft entities.

Record Type: 24000022

Supported Entity Categories: Tanker Aircraft

Table 17. DtTankerAircraftLightingRecord Contents

Field Size (bits)	Tanker Aircraft Lighting Record Fields	
32	Record Type (24000022)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Version	MAK INTERNAL. 16-bit unsigned integer
8	Refueling Station 1 Lights On	8-bit boolean
8	Refueling Station 1 Lights Color	8-bit enumeration
8	Refueling Station 1 Lights Flashing	8-bit boolean
8	Refueling Station 1 DayNight	8-bit enumeration
8	Refueling Station 2 Lights On	8-bit boolean
8	Refueling Station 2 Lights Color	8-bit enumeration
8	Refueling Station 2 Lights Flashing	8-bit boolean
8	Refueling Station 2 DayNight	8-bit enumeration
8	Refueling Station 3 Lights On	8-bit boolean
8	Refueling Station 3 Lights Color	8-bit enumeration
8	Refueling Station 3 Lights Flashing	8-bit boolean
8	Refueling Station 3 DayNight	8-bit enumeration

DtGroundVehicleLightingRecord

This Attribute record adds additional lighting for ground vehicle entities.

Record Type: 24000023

Supported Entity Categories: Ground Vehicles

Table 18. DtGroundVehicleLightingRecord Contents

Field Size (bits)	Ground Vehicle Lighting Record Fields	
32	Record Type (24000023)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Version	MAK INTERNAL. 16-bit unsigned integer
8	Emergency Lights On	8-bit boolean
8	Left Indicator Lights On	8-bit boolean
8	Right Indicator Lights On	8-bit boolean
40	Padding	Padding to 64-bit boundary

DtAisDataRecord

This Attribute record contains the AIS data being transmitted by a surface entity.

Record Type: 24000024

Supported Entity Categories: Surface Vehicles

Table 19. DtAisDataRecord Contents

Field Size (bits)	AIS Data Record Fields	
32	Record Type (24000024)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Version	MAK INTERNAL. 16-bit unsigned integer
32	AIS MMSI	32-bit unsigned integer
32	AIS IMO	32-bit unsigned integer
32	AIS ETA	EpochTime
32	AIS Status	32-bit enumeration
X	AIS Destination	Null Terminated ASCII String

The length of the AIS Destination string can be calculated by subtracting the fixed fields from the Record Length value. String will end with null character and be padded to 64-bit boundary

HLA

In HLA, the NETN FOM adds several subclasses to extend the RPR FOM PhysicalEntity object classes. These include:

- NETN_Aircraft
- NETN_AmphibiousVehicle
- NETN_GroundVehicle

- NETN_MultiDomainPlatform
- NETN_Spacecraft
- NETN_SubmersibleVessel
- NETN_SurfaceVessel
- NETN_Human
- NETN_NonHuman

Please see the NETN FOM for more information on these objects and their attributes.

MAK further subclasses some of the NETN and RPR FOM classes to add further attributes. The *MAK-Physical-2_evolved.xml* FOM module provides the details on these classes and their attributes. These classes include:

- MAK_Aircraft
- MAK_GroundVehicle
- MAK_SurfaceVessel
- MAK_Expendables
- MAK_Radio
- MAK_Sensor
- MAK_Supplies

2.4.8 Radar

MAK has extended electromagnetic emissions to add additional information about radar beams.

DIS

In DIS, a new Attribute record has been added. This is sent as part of an Attribute PDU coupled with the Electromagnetic Emission PDU.

DtRadarBeamExtensionRecord

Record Type: 24000020

Table 20. DtRadarBeamExtensionRecord Contents

Field Size (bits)	Radar Beam Extension Record Fields	
32	Record Type (24000020)	32-bit enumeration
16	Record Length	16-bit unsigned integer
8	PRF Jitter	8-bit boolean
32	Beam Width	32-bit floating point
32	Scan Data Type	32-bit enumeration

Table 20. DtRadarBeamExtensionRecord Contents (continued)

Field Size (bits)	Radar Beam Extension Record Fields	
X	Scan Data	Scan Data Variant
The size and data contained in the Scan Data field is determined by the value of the Scan Data Type enumeration: Unknown (0), Circle (1), Sector (2), Raster (3), Spiral (4), NotScanning (5).		
Unknown and NotScanning have no additional data.		
Circle and Sector data consist of a single 32-bit floating point number.		
Raster data consists of an unsigned 32-bit integer followed by a 32-bit floating point.		
Spiral data consists of two 32-bit floating point numbers.		
Record will be padded to 64-bit boundary.		

HLA

In HLA, a new subclass of the RPR FOM RadarBeam class has been added called MAK_RadarBeam. This class adds new attributes providing additional data on a specific radar beam. For full details on this class, please see the *MAK-DER-1_evolved.xml* FOM module.

2.4.9 Sensor Field of View

MAK has added all new classes to support the transmission of data related to the field of view of a sensor, typically a camera. This is used by VR-Forces to stimulate the gimbaled sensors in the VR-Forces GUI and VR-Vantage.

The easiest way to send and receive this data is by using the remote control toolkit.

DIS

In DIS, two new PDUs are defined: Rectangular Pyramid (below) and Control (page 49).

Sensor Field of View Rectangular Pyramid PDU

This describes a sensor field of view that can be visualized as a rectangular pyramid emanating from the sensor.

PDU Kind: 154

Describes the shape of a sensor field of view that can be visualized as a rectangular pyramid emanating from the sensor. Please see the VR-Forces remote control toolkit for more information on how to publish and reflect data for this PDU.

Table 21. Sensor Field of View Rectangular Pyramid PDU

Field Size (bits)	Field	Description
96	PDU Header	Standard DIS PDU header (uses PDU kind 154)
96	Relative Orientation	3x 32-bit floating-point angles Euler angles specifying orientation relative to host. Only used for Tracking Mode 0.

Table 21. Sensor Field of View Rectangular Pyramid PDU (continued)

Field Size (bits)	Field	Description
192	Relative Location	3x 64-bit floating-point numbers Relative location of the sensor on the host
48	Host Entity ID	Entity ID to which the sensor view is attached
8	Tracking Mode	Enumeration • 0 = Fixed (pointed in fixed direction) • 1 = Entity (tracking a specific target entity) • 2 = Location (tracking a specific location) • 3 = Scan mode (scanning back and forth) • 4 = Manual control mode (requires additional PDUs for control)
8	Version	Integer Version number for this PDU (1)
64	Maximum Zoom Level	Floating point number The maximum level the sensor can zoom. Usually 100.
64	Zoom Level	Floating point number The current zoom level. Between 1 and Maximum Zoom Level.
64	Radius	Floating point number (m) Radius to which the field of view extends (distance from entity). Mainly for visual purposes. 0.0 is the default, and it extends indefinitely.
64	Minimum Azimuth Angle	Floating point angle (radians) The minimum azimuth angle the sensor can rotate to
64	Maximum Azimuth Angle	Floating point angle (radians) The maximum azimuth angle the sensor can rotate to
64	Minimum Elevation Angle	Floating point angle (radians) The minimum elevation angle the sensor can rotate to
64	Maximum Elevation Angle	Floating point angle (radians) The maximum elevation angle the sensor can rotate to
192	Targeted Location	3x 64-bit floating-point numbers Geocentric location being targeted. Only used for Tracking Mode 2.

Table 21. Sensor Field of View Rectangular Pyramid PDU (continued)

Field Size (bits)	Field	Description
32	Color	RGBA color value For visual purposes.
48	Targeted Entity	Entity ID of the target being tracked. Only used for Tracking Mode 1.
512	ID	Unique ASCII name for this sensor, to deconflict when an entity publishes more than one sensor field of view.
64	X Half Angle	Floating point angle (radians) The azimuth angle from the center of the field of view to the edge.
64	Y Half Angle	Floating point angle (radians) The elevation angle from the center of the field of view to the edge.

Sensor Field of View Control PDU

PDU Kind: 156

This PDU is used to send commands to VR-Forces to control the behavior of a sensor field of view. For instance, it can be used to aim and zoom the sensor. Please see the VR-Forces remote control toolkit for more information on how to publish and reflect data for this PDU.

HLA

In HLA, there are three additional classes defined in the *MAK-VRFExt-9.xml* FOM module pertaining to the sensor field of view. The *SensorFieldOfView* object class is a base class that is not intended to be instantiated on its own. Its only subclass, *RectangularPyramid*, describes the shape of a sensor field of view that can be visualized as a rectangular pyramid emanating from the sensor. The *SensorFieldOfViewControl* interaction class can be used to send commands to VR-Forces to control the behavior of a sensor field of view. For instance, it can be used to aim and zoom the sensor. Please see the VR-Forces remote control toolkit for more information on how to publish and reflect data for these classes.

2.4.10 Time and Date

The MAK Time and Date extension is designed to transmit the simulated Time and Date for a simulation. It adds a single object that includes the Time and Date, as well as the Time Advancement Speed, to indicate how fast simulation time is increasing relative to real-world time. An additional attribute is added to indicate whether the sun position should be based on the date and time of day, or if it should be based on the observer position.

For any given exercise, there should be only one Time and Date object to eliminate any conflicts in Date or Time. The simulator updating the Time and Date object will set the Simulation Identifier field to its Site and Host values, leaving entity number to be 0. Since

Time is constantly moving, the Time and Date object should only be updated if the Time of Day Advancement Speed or Sun Position Locked To Observer fields change in value, or on a heartbeat of 60 seconds.

The Time of Day Advancement Speed value is a multiple of real time, indicating that the simulator updating this object is running at some scale of real time.

DIS

In DIS, a custom Time and Date PDU has been created to transmit this information.

Time and Date PDU

PDU Kind: 224

Table 22. Time and Date PDU Contents

Field Size (bits)	MAK Time and Date PDU Fields	
96	PDU Header	Protocol Version. 8-bit enumeration
		Exercise ID. 8-bit unsigned integer
		PDU Type = 224. 8-bit enumeration
		Protocol Family. 8-bit enumeration
		Timestamp. 32-bit unsigned integer
		Length. 16-bit unsigned integer
		PDU Status. 8-bit record
		Padding. 8 bits unused
16	Version	Final PDU Status. Most significant bit MAK INTERNAL. 15-bit unsigned integer
48	Simulation Identifier	Site Number. 16-bit unsigned integer
		Application Number. 16-bit unsigned integer
		Entity Number = 0. 16-bit unsigned integer
64	Time And Date	32-bit integer
		32-bit unsigned integer
64	Time of Day Advancement Speed	64-bit floating point
8	Sun Position Locked to Observer	8-bit boolean

HLA

In HLA, a new object class has been added called MAK_TimeAndDate. For full details on this class, please see the *MAK-VRLExt-3_evolved.xml* FOM module.

2.4.11 Weather

The MAK network representation for weather is based on the NETN Meteorological and Oceanographic (METOC) FOM module. In addition, MAK has developed subclasses of the METOC object classes which extend METOC to add functionality not supported by the standard module. MAK ONE applications are able to reflect the state from both remotely published METOC FOM classes and the MAK METOC extensions.

For DIS support, MAK has designed a custom PDU to include the same data defined in the NETN METOC classes and the MAK extensions.

For both HLA and DIS, VR-Link provides a convenient API for sending and receiving METOC data. The easiest way to interoperate with the MAK weather model in remote applications is to utilize the VR-Link API. However, it is also possible to write your own HLA and DIS interfaces to send and receive METOC data.

VR-Forces can simulate both global and local weather by publishing the MAK METOC extension classes. By default, a single sim engine is chosen to publish global weather conditions. Scenarios can then additionally create local weather areas to change the weather conditions in specific locations. Local weather areas can be created and simulated on any sim engine. However, it is also possible to publish weather state from an external application and VR-Forces will be able to receive, process, and use this data as well. If you choose to develop your own weather server, it is your choice whether to use only the standard NETN METOC object model or the MAK extensions to take advantage of the additional capabilities added.

Note that MAK does not support all capabilities of METOC and does differ from the NETN METOC implementation in some cases.

Differences Between NETN METOC and MAK METOC Object Models

This section describes the differences between the MAK weather model and the standard NETN METOC model. Because the NETN FOM object model supports only HLA, this section focuses on the HLA FOM modules in most of its descriptions. However, the same concepts apply to DIS as well.

Priority

The MAK weather model uses a priority scheme to resolve differences between overlapping weather areas. This is different than how the METOC standard handles area overlap. The METOC standard has rules for calculating the weather state for a location within multiple weather object areas with different values. Each attribute is calculated in different ways, as described in the FOM. This is not supported in MAK ONE products. Instead, each MAK METOC subclass adds a Priority attribute. The values from the weather area with the highest priority will be used. This makes it easier for scenario developers to understand what the weather is at a given location as they define their weather areas. A lower Priority attribute value indicates a higher priority. The global weather is always given the lowest priority (highest value).

If a MAK ONE application receives a METOC object that does not use one of the MAK subclasses, or if it never receives an update including the Priority attribute, a default priority is assigned. This priority will be higher than the priority of VR-Forces global weather, but it will be lower than the priority of most other objects.

GeoReference

Each METOC weather object includes a GeoReference attribute. The GeoReference specifies the area of the world where this weather is valid. The GeoReference has several different variants in METOC. MAK currently supports only three variants of GeoReference:

- Unspecified. If no update is ever received for GeoReference attribute (or it has a discriminant value of 0), this indicates global weather.
- Polygon
- Circle

MAK ONE applications are unable to process any other variants and ignore any weather objects that use them. Another limitation of the MAK implementation is that, once an object is discovered with a particular GeoReference variant, it does not support changing the type of variant. The actual geography of the area can change, but you cannot, for instance, change an area from a polygon to a circle.

Objects in a Weather Area

Each VR-Forces published weather area, including the global weather, results in the creation of multiple objects. For each weather area, VR-Forces creates:

- One MAK_LandSurface object
- One MAK_WaterSurface object
- One or more MAK_TroposphereLayer objects
- One or more MAK_SubsurfaceLayer objects
- One EnvironmentProcess object

Additionally, each VR-Forces sim engine that creates a weather object also creates a single METOC Service object. This one object will be referenced by the ServiceId attribute of all other METOC objects created by this sim engine.

VR-Forces generates so many objects because each weather area includes a full set of weather state, from the subsurface ocean conditions to the clouds in the troposphere. METOC allows the publishing of individual layers independent of other layers, but VR-Forces does not do this. All objects in the same area will have identical GeoReference values.

As a result, VR-Forces tries to associate any weather objects with identical GeoReference values into a single weather area. When an external application creates two independent weather objects with identical GeoReference values, internally VR-Forces associates them. If the GeoReference is updated for one of these objects, VR-Forces assumes that the GeoReference for all objects in that area will be updated as well. If you do not wish weather objects to be associated in this way, make sure they are not given identical GeoReference values.

Overlapping Layers

If multiple SubsurfaceLayers in a given area overlap, preference is always given to the higher layer. For example, if one layer extends from 0-10 feet and another extends from 8-20 feet, VR-Forces will consider it as two layers extending from 0-10 feet and 10-20 feet.

For TroposphereLayers, the same logic applies, but in that case the lower layer is given preference. Thus, in both cases, the layer closer to sea level is given priority.

Incomplete Troposphere or Subsurface Data

The METOC FOM allows only one layer of the troposphere or subsurface to be created in a geographic area. For instance, a particular area can only have troposphere data specified for 1000 to 2000 meters of altitude, with no other layers defined. This is not supported by VR-Forces. VR-Forces always publishes the data for the entire troposphere and the entire subsurface.

If you want to create troposphere or subsurface data to be used by VR-Forces, you should also publish all layers. If data is not provided for a particular altitude or depth in an area where data is provided for other altitudes or depths, then default values will be assumed.

EnvironmentProcess

The EnvironmentProcess created by VR-Forces for a weather area is used only to manage the area. It is an implementation detail, allowing a representative graphic to be added to the GUI. It allows user interaction with the area so that they can modify or delete it. It can generally be ignored by external applications. It contains no weather state.

Disabling VR-Forces Global Weather

If you want to generate global weather state from your own weather server application, you must first disable the global weather created by VR-Forces.

When a new scenario is created, the VR-Forces GUI selects a sim engine to function as the global weather server. You can disable this in the advanced options of the scenario creation dialog. See the "Creating a Scenario" chapter of the *VR-Forces User Guide* or help for details.

Attribute Support

MAK does not support every METOC FOM attribute. The following tables describe the degree to which each METOC attribute is supported. For detailed descriptions for each attribute in the NETN METOC and MAK METOC FOM modules, please see the semantics in the FOM modules and the VR-Link documentation.

Table 23. TroposphereLayer

Attribute	VR-Forces Published	MAK Reflected
Temperature	The LandSurface object and all TroposphereLayer objects are set with the same ambient air temperature. This is the reference temperature for the altitude specified in the MAK_LandSurface attribute ReferenceTemperatureAltitude. Temperature for a given altitude can then be calculated using the International Standard Atmosphere (ISA) model.	Ignored. Reflected temperature will be taken from the LandSurface for a given area. If MAK_Landsurface is published, the ReferenceTemperatureAltitude will be used to calculate the temperate at a given altitude based on the ISA model.
Wind	The HorizontalSpeed and Direction are supported. VerticalSpeed is always set to 0.	The HorizontalSpeed and Direction are supported. VerticalSpeed is ignored.
Precipitation	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same precipitation values.	Not supported. Precipitation for a given area is read from the LandSurface object.
Haze	The haze Type is specified, but the Density is always set to 0.	The haze Type is read, but the Density is ignored.
Humidity	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same humidity value. It represents humidity at sea level.	Not supported. Humidity for a given area is read from the LandSurface object.
BarometricPressure	The LandSurface, WaterSurface, and all TroposphereLayer objects are set with the same ambient air pressure. This is the reference pressure for the altitude specified in the MAK_LandSurface attribute ReferencePressureAltitude. Pressure for a given altitude can then be calculated using the ISA model.	Not supported. Pressure for a given area is read from the LandSurface object.
VisibilityRange	Supported.	Supported.
Cloud	Supported.	Supported.

Table 24. LandSurface

Attribute	VR-Forces Published	MAK Reflected
Temperature	The LandSurface object and all TroposphereLayer objects are set with the same ambient air temperature. This is the reference temperature for the altitude specified in the MAK_LandSurface attribute ReferenceTemperatureAltitude. Temperature for a given altitude can then be calculated using the ISA model.	Used as the temperature for all altitudes in this area. If MAK_LandSurface is published, the ReferenceTemperatureAltitude will be used to calculate the temperate at a given altitude based on the ISA model.
Wind	The HorizontalSpeed and Direction are supported. VerticalSpeed is always set to 0.	Not supported. Wind is for a given area is read from TroposphereLayer objects.
Precipitation	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same precipitation values.	Used as the precipitation for all altitudes in this area.
Haze	The haze Type is specified, but the Density is always set to 0.	Not supported. Haze for a given area is read from the TroposphereLayer objects.
Humidity	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same humidity value. It represents humidity at sea level.	Used as the humidity for all altitudes in this area.
BarometricPressure	The LandSurface, WaterSurface, and all TroposphereLayer objects are set with the same ambient air pressure. This is the reference pressure for the altitude specified in the MAK_LandSurface attribute ReferencePressureAltitude. Pressure for a given altitude can then be calculated using the ISA model.	Used as the pressure for all altitudes in this area. If MAK_LandSurface is published, the ReferencePressureAltitude will be used to calculate the pressure at a given altitude based on the ISA model.
VisibilityRange	Supported.	Not supported. VisibilityRange for a given area is read from TroposphereLayer objects.

Table 24. LandSurface (continued)

Attribute	VR-Forces Published	MAK Reflected
Snow	Supported. Additional state regarding snow at elevation and on roads is specified in the MAK_LandSurface attributes SnowMinimumAltitude and SnowRoadAccumulation.	Supported. If MAK_LandSurface is published, the SnowMinimumAltitude and SnowRoadAccumulation will be used to specify snow accumulation at altitudes and on roads.
Moisture	Only dry and wet conditions are supported.	Only dry and wet conditions are supported.
IceCondition	Not supported.	Not supported.

Table 25. WaterSurface

Attribute	VR-Forces Published	MAK Reflected
Temperature	Not supported.	Not supported.
Wind	The HorizontalSpeed and Direction are supported. VerticalSpeed is always set to 0.	Not supported. Wind for a given area is read from TroposphereLayer objects.
Precipitation	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same precipitation values.	Not supported. Precipitation for a given area is read from the LandSurface object.
Haze	The haze Type is specified, but the Density is always set to 0.	Not supported. Haze for a given area is read from the TroposphereLayer objects.
Humidity	The LandSurface, WaterSurface, and all TroposphereLayer objects are published with the same humidity value. It represents humidity at sea level.	Not supported. Humidity for a given area is read from the LandSurface object.

Table 25. WaterSurface (continued)

Attribute	VR-Forces Published	MAK Reflected
BarometricPressure	The LandSurface, WaterSurface, and all TroposphereLayer objects are set with the same ambient air pressure. This is the reference pressure for the altitude specified in the MAK_LandSurface attribute ReferencePressureAltitude. Pressure for a given altitude can then be calculated using the ISA model.	Not supported. Pressure for a given area is read from the LandSurface object.
VisibilityRange	Supported.	Not supported. VisibilityRange for a given area is read from TroposphereLayer objects.
SeaState	Supported.	Supported.
Salinity	Not supported.	Not supported.
Tide	Supported.	Supported.
Ice	Not supported.	Not supported.
Current	Supported.	Supported.
Wave	Length, Height, and Period values will be set based on the current VR-Forces sea state. Direction is supported.	Length, Height, and Period are not supported. Only SeaState is used. Direction is supported.
Swell	Length, Height, and Period values will be set based on the current VR-Forces swell state. Direction is supported.	Swell state is set based on the swell length and height. Period is not supported. Direction is supported.

Table 26. SubsurfaceLayer

Attribute	VR-Forces Published	MAK Reflected
Temperature	All layers in a given area will be set with the same ambient water temperature.	Ambient water temperature for the entire area will be read from the layer closest to the surface.
Current	Not supported	Not supported
Salinity	Not supported	Not supported
BottomType	Not supported	Not supported

Interactions

MAK does not support any of the METOC interaction classes.

DIS

MAK has designed a custom METOC PDU to represent the data in both the NETN METOC object classes as well as MAK's extensions.

The NETN METOC FOM Module defines several object classes: LandSurface, WaterSurface, TroposphereLayer, and SubsurfaceLayer. The MAK-METOC FOM Module extends these classes with MAK_LandSurface, MAK_WaterSurface, MAK_TroposphereLayer, MAK_SubsurfaceLayer, and also a new class, MAK_WindCorridor. The MAK METOC PDU supports all of these object classes.

The METOC PDU can represent any of the METOC FOM object classes. The data fields common throughout most of the METOC object classes are part of the fixed body portion of the PDU. Data fields that are specific to only a certain class of METOC object are specified in custom standard variable records, called METOC Layer Records. Each instance of the MAK METOC PDU shall contain only 1 METOC Layer Record.

The MAK METOC PDU is issued by a simulation to indicate a new weather area and condition in the simulated environment. The weather area covers an area specified by the GeoReference field of the PDU. If the GeoReference field is all 0s, the weather area is global. A new MAK METOC PDU is sent anytime the conditions of the weather area change, or on a heartbeat.

For a weather area with different surface conditions and layers, multiple MAK METOC PDUs are issued, one for each surface or layer. All layers and conditions that constitute a single weather area should have identical GeoReference values and VR-Forces Weather Area Identifiers.

The contents of the PDU and the Layer Records it supports are described below.

METOC PDU

PDU Kind: 221

Table 27. METOC PDU Contents

Field Size (bits)	MAK METOC PDU Fields	
96	PDU Header	Protocol Version. 8-bit enumeration
		Exercise ID. 8-bit unsigned integer
		PDU Type = 221. 8-bit enumeration
		Protocol Family. 8-bit enumeration
		Timestamp. 32-bit unsigned integer
		Length. 16-bit unsigned integer
		PDU Status. 8-bit record
		Padding. 8 bits unused
16	Version	Final PDU Status. Most significant bit
		MAK INTERNAL. 15-bit unsigned integer

Table 27. METOC PDU Contents (continued)

Field Size (bits)	MAK METOC PDU Fields	
48	Entity Identifier	Site Number. 16-bit unsigned integer
		Application Number. 16-bit unsigned integer
		Entity Number. 16-bit unsigned integer
128	Unique ID	Universally Unique Identifier. 128 bits
32	Temperature	32-bit floating point
96	Wind	Direction. 32-bit floating point
		Horizontal Speed. 32-bit floating point
		Vertical Speed. 32-bit floating point
64	Precipitation	Type. 32-bit enumeration
		Intensity. 32-bit float
64	Haze	Type. 32-bit enumeration
		Density. 32-bit float
32	Humidity	32-bit floating point
32	Barometric Pressure	32-bit floating point
32	Visibility Range	32-bit floating point
32-?	GeoReference Record	Type. 32-bit enumeration
		Length. 32-bit unsigned integer
		Data
	Layer Record	Type. 32-bit enumeration
		Length. 16-bit enumeration
		Data

Land Surface Record

Record Type: 24000101

Table 28. Land Surface Record Contents

Field Size (bits)	Land Surface Record Fields	
32	Record Type (24000101)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Priority	16-bit unsigned integer
128	VR-Forces Weather Area ID	Universally Unique Identifier. 128 bits
64	Snow	Depth. 32-bit floating point
		Density. 32-bit floating point
16	Moisture	16-bit enumeration
16	Ice Condition	16-bit enumeration

Table 28. Land Surface Record Contents (continued)

Field Size (bits)	Land Surface Record Fields	
32	Reference Temperature Altitude	32-bit floating point
32	Reference Pressure Altitude	32-bit floating point
32	Rain Accumulation	32-bit floating point
64	Snow Minimum Altitude	64-bit floating point
64	Snow Road Accumulation	64-bit floating point
32	Visual Obscurant Color	32-bit unsigned integer
32	Visual Obscurant Height	32-bit floating point
8	Is Lightning Present	8-bit boolean
32	Padding	32 bits padding

Water Surface Record

Record Type: 24000102

Table 29. Water Surface Record Contents

Field Size (bits)	Water Surface Record Fields	
32	Record Type (24000102)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Priority	16-bit unsigned integer
128	VR-Forces Weather Area ID	Universally Unique Identifier. 128 bits
16	Sea State	16-bit enumeration
32	Salinity	32-bit floating point
32	Tide	32-bit floating point
80	Ice	Type. 16-bit enumeration
		Thickness. 32-bit floating point
		Coverage. 32-bit floating point
64	Current	Direction. 32-bit floating point
		Speed. 32-bit floating point
128	Wave	Length. 32-bit floating point
		Height. 32-bit floating point
		Period. 32-bit integer
		Direction. 32-bit floating point
128	Swell	Length. 32-bit floating point
		Height. 32-bit floating point
		Period. 32-bit integer
		Direction. 32-bit floating point

Table 29. Water Surface Record Contents (continued)

Field Size (bits)	Water Surface Record Fields	
32	Choppiness	32-bit floating point
32	Surge Depth	32-bit floating point
32	Surface Transparency	32-bit floating point
8	Drive Sea State from Wind	8-bit boolean

Troposphere Layer Record

Record Type: 24000103

Table 30. Troposphere Layer Record Contents

Field Size (bits)	Troposphere Layer Record Fields	
32	Record Type (24000103)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Priority	16-bit unsigned integer
128	VR-Forces Weather Area ID	Universally Unique Identifier. 128 bits
64	Layer	Base. 32-bit floating point
		Thickness. 32-bit floating point
96	Cloud	Type. 32-bit enumeration
		Coverage. 32-bit floating point
		Density. 32-bit floating point

Subsurface Layer Record

Record Type: 24000104

Table 31. Subsurface Layer Record Contents

Field Size (bits)	Subsurface Layer Record Fields	
32	Record Type (24000104)	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Priority	16-bit unsigned integer
128	VR-Forces Weather Area ID	Universally Unique Identifier. 128 bits
64	Layer	Base. 32-bit floating point
		Thickness. 32-bit floating point
64	Current	Direction. 32-bit floating point
		Speed. 32-bit floating point
32	Salinity	32-bit floating point
32	Bottom Type	32-bit enumeration
32	Permeability	32-bit floating point

Wind Corridor

Record Type: 24000105

Table 32. Wind Corridor Contents

Field Size (bits)	WindCorridor Layer Record Fields	
32	Record Type	32-bit enumeration
16	Record Length	16-bit unsigned integer
16	Priority	16-bit unsigned integer
128	VR-Forces Weather Area ID	Universally Unique Identifier. 128 bits
32	Width	32-bit floating point
32	Height	32-bit floating point
32	Wind Speed	32-bit floating point
X	Path	Array of World Coordinate Positions

The length of the Path can be calculated by subtracting the size of the fixed fields (Record Type, Record Length, Priority, VR-Forces Weather Area ID, Width, Height and Wind Speed) from the overall record length. Then divide that by the size of a World Coordinate (3 64-bit floats) to get the number of positions in the Path array.

HLA

In HLA, MAK supports the object classes defined in the NETN METOC FOM module as well as the additional classes defined in the MAK METOC extensions. Please see the *NETN-METOC.xml* and *MAK-METOC-1_evolved.xml* FOM modules for more details.



3. VR-Forces External Communication Server Interface

This chapter details the message formats that VR-Forces uses to interface to an external communication effects server when this feature is enabled. These messages are custom fields within HLA RPR FOM or DIS messages.

3.1. Overview	64
3.1.1 Messages From VR-Forces Sim	64
3.1.2 Messages To VR-Forces Sim	64
3.1.3 Ready To Send Mode	64
3.2. Request Message	65
3.2.1 Signal Data String	65
3.3. Response Messages	66
3.3.1 Process Message	66
3.3.2 Timeout	67
3.4. Ready To Send Signal (RTSS)	67

3.1. Overview

In this mode, VR-Forces will wait for responses from the external communication effects server any time a simulated radio message is sent from one entity to another. VR-Forces will pass the radio message to the receiving entities only when the communication effects server indicates, and with any delays that are indicated by the communication effects server.

It is assumed that the external communication effects server is simulating the radio network connectivity, possibly by using published RPR/DIS Transmitter state, or using any other internal model of the communication for the scenario.

3.1.1 Messages From VR-Forces Sim

When an entity sends a radio message, VR-Forces sends a communication effects request using a Radio Signal Interaction on the network that includes additional data to be received by the communication effects server. This signal interaction initiates communication effects processing for a specific message. If VR-Forces is configured to publish radio transmitter state objects, then all transmitter and receiver entity IDs are the IDs of the transmitters. Otherwise, these are the IDs of the transmitting/receiving entities.

3.1.2 Messages To VR-Forces Sim

VR-Forces listens for three specific response messages from the communication effects server, all of which are custom Data Interactions:

- **Process Message:** Indicates that a message should be received by one or more recipients.
- **Timeout:** Indicates that processing is complete for a specific message and no more radios will be receiving that message.
- **Ready To Send Signal (RTSS):** Optional message that indicates a specific radio in VR-Forces can be put in the “ready to send” state, and is allowed to send a message.

3.1.3 Ready To Send Mode

This is an optional mode that can be configured in VR-Forces by setting the value of `use-ready-to-send` in the `commModelParams.mtl` file in the simulation model set to `True`. This is off by default.

When this mode is on, VR-Forces assumes that the external communication effects server is modeling how long a radio takes to transmit a message and that only one message can be transmitted at a time. In this mode, each time a VR-Forces radio sends a message, that radio’s ready-to-send state is toggled to false, and is only reset to true when VR-Forces receives an RTSS message from the external communications effects server. Any messages that are requested to be sent by a radio that is not ready to send are queued. Once the ready-to-send state is reset to true, the next message in the queue is sent.

When this mode is off, VR-Forces will send radio messages without any delay and no Ready To Send Signal messages are expected by VR-Forces.

3.2. Request Message

The RPR FOM or DIS standard Application Specific Signal Interaction is used, with a custom string in the SignalData field to request communication effects modeling for a specific message.

The Entity Identifier Record in the signal interaction must be either:

- The identifier of the radio transmitter associated with the message, if a radio transmitter is being published.
- The identifier of the transmitting entity, if no radio transmitter is being published.

All signal interactions encoded with external communication effects server data will use the ProtocolID of 10000.

In DIS, the ProtocolID will be encoded into the first 4 bytes of the signal data.

The signal data will contain a string formatted as detailed below. This string may be followed by arbitrary additional data that is not intended for the external communication effects server.

3.2.1 Signal Data String

The signal data string starts with the string HEADER and a newline, followed by attribute-value pairs, each followed by a newline, and ends with the string EOH and a newline. Attribute-value pairs are from the table below, and can be in any order.

All values are specified in ASCII.

```
HEADER
<attribute>=<value>
<attribute>=<value>
...
EOH
```

Table 33. Signal data attribute-value pairs

Name	Description	Value	Examples
receiver	Destination Entity ID (only for unicast messages). Optional.	3 integers separated by periods	receiver=1.1.1001 receiver=2.3.1005
size	Size of the message to simulate, in bytes or seconds.	Integer bytes or float seconds.	size=4000 bytes size=10 seconds
timeout	Number of wall-clock seconds VR-Forces will wait for a response to this message before discarding it.	Floating point number	timeout=180
timestamp	Unique ID for this message.	32-bit value represented in hexadecimal	timestamp=0x0A4B22E1

Note that:

- The size field has no relation to the actual data payload on the network. This is the size of the message to be simulated. It may be in bytes for digital messages, or in seconds for audio stream type messages.
- The timestamp field is the ID that links requests to responses from the communication effects server. It does not need to reflect actual time advancement and must be unique for each message.

3.3. Response Messages

All responses from the communication effects server are in Data Interactions.

Each response type has a unique DatumID value.

Variable Datum entries should not exceed 1000 bytes. Multiple interactions should be sent if there is more data than can fit in one interaction.

For all responses, the OriginatingEntity field of the data interaction should be the entity ID of the entity that sent the original signal interaction request.

3.3.1 Process Message

Process Message interaction is sent by the external communication effects server to indicate that one or more entities has successfully received a message.

The Variable Datum should be formatted as follows:

DatumID: 60001

DatumValue:

Table 34. Process message interaction values

Field	Description	Type	Size
Source Entity ID	ID of entity or transmitter sending the message; originally sent in the Request message.	Entity ID	6 bytes
Transmitter radio index	Radio index of the transmitting radio; originally sent in the Request message.	UInt16	2 bytes
Timestamp / Message ID	Message ID matching the timestamp field in the Request message.	32 bit arbitrary data	4 bytes
Number of receiving entities	Number of receivers included in this message	UInt32	4 bytes
Receiver 1 ID	ID of entity to receive the message.	Entity ID	6 bytes
Receiver 1 Delay	Delay in seconds.	Float64	8 bytes
Receiver 2 ID	ID of entity to receive the message.	Entity ID	4 bytes
Receiver 2 Delay	Delay in seconds.	Float64	8 bytes
...	...	...	...

Note that:

- The *delay* value indicates the number of seconds after message transmission that the receiver receives the message.
- VR-Forces does not currently use the delay value. If the delay is 0 or greater, VR-Forces processes the message for the receiver at the time the Process Message is received. It is assumed that the Process Message is sent at a time consistent with the delay value.

3.3.2 Timeout

Timeout interaction is sent by the external communication effects server to indicate that the processing of a specific message is complete, and that there will be no further Process Message interactions sent with the indicated message timestamp.

DatumID: 60002

DatumValue:

Table 35. Timeout interaction values

Field	Description	Type	Size
Source Entity ID	ID of entity or transmitter sending the message; originally sent in the Request message.	Entity ID	6 bytes
Transmitter radio index	Radio index of the sending radio; originally sent in the Request message.	Uint16	2 bytes
Timestamp / Message ID	Message ID matching the timestamp field in the Request message.	32 bit arbitrary data	4 bytes
Number of packets	Unused.	Uint32	4 bytes
Number of entities	Unused.	Uint32	4 bytes

Note that:

- VR-Forces does not use the Number of Packets or Number of Entities field but does parse these fields, so they must be present in the data.
- Additional data can be included following the message, but VR-Forces ignores it.

3.4. Ready To Send Signal (RTSS)

The RTSS interaction is sent by the external communication effects server to indicate that a transmitter is allowed to send a new message. RTSS is optional and is only expected by VR-Forces if the use-ready-to-send mode is enabled.

DatumID: 60010

DatumValue:

Table 36. RTSS interaction values

Field	Description	Type	Size
Source Entity ID	ID of entity or transmitter sending the message; originally sent in the Request message.	Entity ID	6 bytes
Transmitter radio index	Radio index of the sending radio; originally sent in the Request message.	Uint16	2 bytes
Timestamp / Message ID	Unused	32-bit arbitrary data	4 bytes
Window Time	Unused	Uint32	4 bytes

Note that VR-Forces does not use:

- The Message ID.
- Window Time.



4. Default Ports for MAK ONE Products

This chapter lists the default ports for MAK ONE products and their components.

4.1. MAK Legion	70
4.2. MAK Data Logger	70
4.3. MAK RTI	70
4.4. VR-Link	70
4.5. VR-Exchange	71
4.6. WebLVC	71
4.7. VR-Vantage	71
4.8. VR-Forces	71
4.9. VR-Engage	72
4.10. RFView for MAK ONE	72
4.11. RadarFX	72
4.12. FLEXlm Licensing	73
4.13. Tracy	73
4.14. Loopback Connections	73

4.1. MAK Legion

Port 6000 is the default port.

Table 37. MAK Legion Port Value

Default Port	To Use a Different Port...
6000	Change in command line or relevant config file (<code>./bin64/legionMonitor.lua</code> , <code>./bin64/legionServer.lua</code>)

4.2. MAK Data Logger

Table 38. MAK Data Logger Port Values

Default Port	Protocol / Usage	To Use a Different Port...
3000	DIS	Change in GUI, command line, or config file
4053	UDP: Remote Control	Change in command line or config file
5000	UDP: Video Streaming	Change in GUI or config file

4.3. MAK RTI

The MAK RTI ports are Modified in the RTI GUI or in *rid.mtl*. This information applies only to HLA connection. To learn more about the default ports for the MAK RTI, refer to the *MAK RTI User Guide* or help.

Table 39. MAK RTI Port Values

Default Port	Protocol / Usage	To Use a Different Port...
4000	UDP Port	Edit <i>rid.mtl</i> (setqb RTI_udpPort 4000)
4001	TCP Port	Edit <i>rid.mtl</i> (setqb RTI_tcpPort 4001)
5000	TCP: rtiForwarder	Edit <i>rid.mtl</i> (setqb RTI_distributedForwarderPort 5000)
6001	Multicast Discovery Port	RTI_mcastDiscoveryPort
6003	UDP: RTI Assistant Port	RTI_ASSISTANT_PORT (environment variable)
15164	Federate Protocol Server	RTI_FEDPRO_PORT (environment variable)

4.4. VR-Link

Port 3000 is the default for all DIS protocols. For HLA, see MAK RTI.

Table 40. VR-Link Port Value

Default Port	Protocol / Usage	To Use a Different Port...
3000	DIS	Change in command line

4.5. VR-Exchange

Table 41. VR-Exchange Port Values

Default Port	Protocol / Usage	To Use a Different Port...
3000	UDP: DIS Broker	Change in GUI, command line, or config file
5111	TCP: Brokers to heartbeat the portal	Change in GUI or config file

4.6. WebLVC

Table 42. WebLVC Port Values

Default Port	Protocol / Usage	To Use a Different Port...
8080	Port for Proxy	Change in GUI or config file
9101	Web Server Port	Change in GUI or config file
9102	TCP Server Port	Change in GUI or config file

4.7. VR-Vantage

Table 43. VR-Vantage Port Values

Default Port	Protocol / Usage	To Use a Different Port...
3000	DIS	Change in GUI or command line
5000	UDP: Video Streaming Port	Change in GUI
22563	TCP: Remote Display Engines	Change in GUI or command line

4.8. VR-Forces

Table 44. VR-Forces Port Values

Default Port	Protocol / Usage	To Use a Different Port...
3000	DIS	Change in GUI or command line
9000	TCP: Transporting scenario files on load and save	Set <code>--fileTransporterReceivePort</code> on the GUI command line

Table 44. VR-Forces Port Values (continued)

Default Port	Protocol / Usage	To Use a Different Port...
Random	Port used by the DtCgfDispatcher's file transporter to receive order of battle and plan files from a DtVrfRemoteController. If the port requested cannot be opened, the next open port will be used. A value of zero indicates that VR-Link should choose an ephemeral port, which is automatically assigned by the system.	The only way to change the port is in the simulation engine. Edit: cgfDispatcherReceivePort in the vrfSim.mtl located in <code>./appData/settings/vrfSim</code> folder (setqb cgfDispatcherReceivePort0)

4.9. VR-Engage

Table 45. VR-Engage Port Value

Default Port	Protocol / Usage	To Use a Different Port...
3000	DIS	Change in GUI or command line

4.10. RFView for MAK ONE

Table 46. RFView for MAK ONE Port Value

Default Port	Protocol / Usage
5132	RFView Sidecar

4.11. RadarFX

Table 47. RadarFX Port Values

Default Port	Protocol / Usage	To Use a Different Port...
4567	Simple Client	Change in GUI
3000	DIS, RadarFX Server	Change in GUI

4.12. FLEXlm Licensing

Table 48. FLEXlm Licensing Port Values

Default Port	To Use a Different Port...
27000	Edit SERVER hostname MAC_Address 27500 (27500 is the defined port)
Defined @ runtime	Set by adding port=xxxx, for example: VENDOR maklmgrd ./maklmgrd port=1234

4.13. Tracy

Table 49. Tracy Port Value

Default Port	Protocol / Usage	To Use a Different Port...
8086	UDP	TRACY_PORT (environment variable)

4.14. Loopback Connections

When using a loopback connection for DIS or Legion, all executables must have the incoming and outgoing connection open for the loopback to work correctly.

► **To set up an inbound rule for a Windows program:**

1. Run Windows Defender Firewall with Advanced Security.
2. Select Inbound Rules.
3. Choose **Action > New Rule**.
4. Select Program and then click Next.
5. Enter the path to the program executable and then click Next.
6. Select Allow the Connection and then click Next.
7. For the rule, select Domain, Private, and Public. Click Next.
8. Optionally, enter a name for the rule.
9. Click Finish.



5. Basic Troubleshooting

This chapter consists of troubleshooting checklists for some of the most common problems MAK customers encounter.

5.1. Product Does Not Run	76
5.1.1 Linux Applications	76
5.1.2 HLA Applications	76
5.1.3 License Management Checklist	77
5.2. MAK ONE Products Do Not Work Together	79
5.2.1 DIS Applications	79
5.2.2 HLA Applications	81
5.3. Using Custom FOM Configurations	84
5.4. Using the DIS Spy	84
5.4.1 Starting the DIS Spy	84
5.4.2 Connecting to a Simulation	84
5.4.3 Viewing PDU Information	84
5.5. Using Older MAK ONE Products with HLA Evolved	85
How to Contact Us	85
Telephone	85
E-mail	85
Internet	85
Post	86

5.1. Product Does Not Run

If you start a MAK ONE product and nothing happens, or a console window flashes, the most likely problem is that license management is not properly configured. See the "5.1.3 License Management Checklist" on the facing page for details.

Most products create a log file in the `./bin64` directory. Check the log for messages. If there is no log, run the executable from the command prompt to check the error messages.

5.1.1 Linux Applications

If you are running an application on Linux, make sure you are running on the same version as that on which the application was compiled (for example, Red Hat Linux Enterprise Workstation 7.0 or 8.0).

5.1.2 HLA Applications

If an HLA application does not start (or the `vpNetHLA` module fails to load), check the following:

- If you get a message that the program cannot find a certain libraries—*libfedtime*, *libRTI-NG*, *libfedtime1516*, or *librti1516*—you are trying to run an HLA executable and the RTI libraries could not be found. Make sure that an RTI is installed on the machine and that the path to the RTI libraries is in your path:
 - `PATH` on win32
 - `LD_LIBRARY_PATH` on Linux and Solaris.
- Make sure that the application can find the FED file. The FED file should be in the directory from which the application is running or at the location specified by the `RTI_CONFIG` environment variable.
- If the federation requires an *rtiexec*, make sure the *rtiexec* is running and that each federate is capable of connecting to it.

MAK RTI

If you are using the MAK RTI and receive one of these error messages at startup, the problem is as indicated:

- ****FATAL ERROR**:** Could not create Federation Execution VR-Link:
RTI exception: CouldNotOpenFED ()
The executable could not find the FED file in either its current directory or the location specified by `RTI_CONFIG`.
- ****FATAL ERROR**:** Could not create Federation Execution VR-Link:
RTI exception: RTIinternalError ()
The executable may be looking for an *rtiexec* that is not running.

See "5.2.2 HLA Applications" on page 81 (under MAK ONE Products Do Not Work Together) for more HLA troubleshooting tips.

5.1.3 License Management Checklist

If your MAK ONE product does not start and you believe that it is a license management problem, check the following:

- Did you read and follow the license management installation instructions in your product's users guide?

We do not mean to be facetious when we ask this question. Many customers obtain MAK ONE products via download and do not realize that the installation includes a PDF version of the product documentation, which includes installation instructions. Every MAK ONE product has a `./doc` directory, which contains user documentation. Installation instructions are typically found in the Installing chapter of the users guide.

- Are you running the correct version of the FLEXlm software?
- Is the license server running?

To check whether the license server is running, on the server machine, run the *lmstat* utility in the *flexlm* directory.

- Do you have a valid license for the product and operating system you are trying to start?
- Does the name in the license file match the name of the license server?
- Is the license file an ASCII text file? We provide the license file in text format, but some customers open it in a word processor such as Microsoft Word to look at it and do not save it back to text format.
- Does the license file have the extension *.lic* and is it in the License Management directory?

i When looking at files in Windows Explorer, make sure that you have not hidden extensions for known file types. This option can be found in Tools/Folder Options/View, and it is often ON by default. Make sure that "Hide extensions for known file types" is NOT checked.

- Did you set the MAKLMGRD_LICENSE_FILE environment variable on the license server and on the computer on which you are trying to start an application (if the computer is different from the license server)?

i For applications built with VR-Link 3.13.1 or later, use of the MAKLMGRD_LICENSE_FILE environment variable is optional. The alternatives for specifying the license server are described in the Installing chapter in the users guide for your MAK ONE product.

- If the computer on which you are running the application is not the same computer as that which is running the license server, can this computer see the license server over the network?

To see if one computer can see another, from a command prompt, run:

```
ping server_name
```

Errors will be reported if the server cannot be found.

- If you have upgraded to a new version of a MAK ONE product, does your maintenance agreement cover this upgrade? (If it does not, you will probably get an error message stating that the license does not support the software.)

If the answer to any of these questions is no, make the needed changes and try again. If the answer to all of these questions is yes, contact MAK support at <https://www.mak.com/mak-one/support/portal>. You may have a corrupt or otherwise invalid license file or some other problem that we can help you with.

License Management Frequently Asked Questions

This section lists some common questions about license management.

- Do I need to start the license server each time I boot the server machine?
The license server must be manually started each time the machine starts up unless it has been configured otherwise.
- How do I configure the MAK license server as a service on Windows?
On Windows, use the *lmtools* utility from the *flexlm* directory. In the Configure Services tab, specify a Service Name, lmgrd path, license file, and log file. Then check Use Services and Start Server at Power Up.
- How can I know that the license server has started successfully on UNIX?
The license server's LOG file (in */usr/local/flexlm*) should list all the components that are in your license file.
- How can I know that the license server has started successfully on Windows?
 - If the license server has been configured as a service, the license server's log file (specified in *lmtools*) should list the components from your license file.
 - If the license server has been started with *runLM*, the console window should list all the components from your license file.
- How can I find out which licenses are currently in use?
On the server machine, run `lmstat -A` from a console.

License Management Error Messages

This section lists common FLEXlm error messages, their causes, and how to respond.

- Cannot find license file [FLEXlm error: -1]
Neither a license file nor a license server could be found. Make sure the MAKLMGRD_LICENSE_FILE environment variable is set.
- Cannot connect to license server [FLEXlm error: -15]
The license server machine could not be found. Either the client machine is not networked to the server machine or you need to set the MAKLMGRD_LICENSE_FILE environment variable.
- License server does not support this version of this feature [FLEXlm error: -25]
The maintenance for this product is not up-to-date in the license file. Send an email to keys@mak.com with your information.

- Licensed number of users already reached [FLEXlm error: -4]
All available licenses are in use for the product. One of the licenses must be checked in before the program will start.
- No such feature exists [FLEXlm error: -5]
Your license file does not have a license for the product you are trying to run. Check your license file. You can obtain licenses from keys@mak.com.
- Feature has expired [FLEXlm error: -10]
Your product license has expired. Contact keys@mak.com to get a new license.
- The FLEXlm License Finder dialog box always appears at startup. This dialog box asks you to specify either the license server or the license file.
Make sure that the license server is running and that MAKLMGRD_LICENSE_FILE is set.

5.2. MAK ONE Products Do Not Work Together

If you are trying to use several MAK ONE products together, such as the Data Logger and VR-Vantage or VR-Forces and VR-Vantage, and you cannot get data from one to be visible in the other, it is usually a network or simulation protocol configuration problem, or you are running incompatible versions of the applications.

5.2.1 DIS Applications

If DIS applications do not see each other:

- Make sure that you are running the DIS versions of the applications.
- Are all applications using the same port?
- Do all machines share a common netmask and are all applications sending data to the same IP broadcast address?
- Can the computers ping each other?
- If you are playing a Data Logger recording, are you playing a recording of a DIS exercise?

DIS FAQ

- Where are the MAK DIS applications?

In older MAK ONE products, DIS applications are usually in the *./binDIS* directory. On Windows, some DIS applications have DIS as part of the product name in the Start menu. For VR-Forces, you choose a DIS connection in the VR-Forces Launcher. The DIS applications are in *./bin64* and the file names include DIS.

 VR-Vantage applications are in *./bin64*. They are not protocol-specific. You choose a simulation connection after you start the base executable.

- How can I set the port?

By default, most MAK ONE products use port 3000. (See "Chapter 4. Default Ports for MAK ONE Products" on page 69 for a list.) To change the port, start an application with the `-P` command-line option. The Logger, VR-Forces, and VR-Vantage let you set the port dynamically as part of the connection process.

Troubleshooting the Subnet Mask

To troubleshoot DIS connection problems, try running two VR-Link F18 applications. If you do not have VR-Link, the VR-Link net utilities installer has simple DIS applications that do not require a license.

To run the F18 DIS application, locate the `f18DIS_64` executable in the `./bin64` directory. Launch two F18DIS applications. To specify a destination broadcast address, use the `-A` argument. To specify a port, use `-P`. To specify a device address use `--deviceAddress`.

The f18DIS applications print their connection settings to the console when they are initialized in the format `DESTINATION_ADDRESS:PORT`, for example:

```
DIS socket OK to 10.255.255.255:3000
```

When the F18 example discovers another F18 on the network, it prints a second New Entity line.

If the F18DIS applications both print that they opened a DIS Socket on the same `DESTINATION_ADDRESS:PORT` combination, but do not see each other or are able to shoot each other down, there is a break in the connection. Specifying the subnet mask for your network will usually fix the issue. This is especially likely if your subnet mask does not fall into the IPv4 classful default values listed in Table 50.

Table 50. IPv4 classful default values

Class	Leading bits	Default subnet mask in dot-decimal notation
Class A	0	255.0.0.0
Class B	10	255.255.0.0
Class C	110	255.255.255.0

To set a subnet mask for the F18 DIS application, use the `--subnetMask` command line argument.

Finding or Changing the Subnet Mask

Typically, machines on a LAN have the correct broadcast addresses, so VR-Link applications will be able to communicate with each other immediately.

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
```

In Windows systems, you can use `ipconfig` to examine the configuration of network devices.

If your netmask appears to be wrong in your network settings, please see your system administrator.

VR-Link computes the broadcast address using the device's netmask and host address. 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.

5.2.2 HLA Applications

If HLA applications do not see each other:

- Make sure you are running the HLA versions of the applications. If you are using ownership management, DDM, or MOM services, make sure you are running applications built against the same RTI specification (1.3 or 1516).
- Is the same version of the RTI installed on each computer on which an application is running?
- Are you using the correct version of the RTI (1.3 or 1516) for your applications?
- If required, did you start the rtiexec?
- Are the RID files configured the same way? For example, if you are using the MAK RTI, is the **udpPort** value the same on each computer?
- Are all federates using the same FED file and can they find the FED file? The FED file should be in the directory from which the application is running or at the location specified by the RTI_CONFIG environment variable.
- Are all federate joining the same federation?
- Can the computers ping each other?
- If you are playing a Data Logger recording, are you playing a recording of an HLA exercise?

HLA FAQ

Many of the configuration issues described in this section are RTI implementation-specific. Assume that the information is specific to the MAK RTI unless specified otherwise.

- Where are the MAK HLA applications?

In older version of MAK ONE products, HLA applications are stored in `./binHLA13` or `./binHLA1516`. The `./binHLA13` directory contains executables for use with RTI NG 1.3. The `./binHLA1516` directory contains HLA 1516 executables. On Windows, HLA applications have either HLA 1.3 or HLA 1516 as part of the product name in the Start menu or the connection dialog box.

On more recent versions of MAK ONE products, HLA applications are in `./bin64`. The version of the application is indicated by the presence of 1516, 1516e, or 4 in the executable name.

 VR-Vantage applications are in `./bin64`. They are not protocol-specific. You choose a simulation connection after you start the base executable.

- How do I use an RTI?

To use an RTI, each HLA application (or federate) must load the RTI libraries. In most cases, the *rtiexec* must also be running.

- How does an HLA application find the RTI libraries?

Typically, the RTI libraries are dynamically linked with the executable. On Windows, this means that an HLA application will look for the libraries in its current directory, and then in the directories specified in the PATH. On IRIX, the executable will look in the directories specified in LD_LIBRARYN32_PATH. On other UNIX-based systems, the executable will look in the directories specified in LD_LIBRARY_PATH.

- Where can I make changes to default RTI settings or view current RTI settings?

RTI configuration settings are in the RTI's RID file. For the MAK RTI, this file is called *rid.mtl* and is located in the RTI's top level directory. You can view, but not edit, RID settings in the RTI Assistant's Federations View. You can also view RID settings in the RTIspy. Some settings can be edited.

- How can I specify which RID file to load?

The easiest way to specify which RID file to load is to use the RTI Chooser. For details, please see the *MAK RTI Users Guide*. Otherwise, the search order for the RID file depends on how you specify it.

If the RTI_RID_FILE environment variable is not defined, the search order is:

1. Try to open *rid.mtl* in the directory from which you run the application (which is not necessarily the directory in which the executable is located.)
2. Try to open the file specified by *RTI_CONFIG/rid.mtl*.
3. If no RID file is identified by one of these methods, the MAK RTI prints a warning and opens using default values.

If the RTI_RID_FILE environment variable is defined, the search order is:

1. Try to open the file specified in RTI_RID_FILE.
2. Try to open the file specified by *RTI_CONFIG/RTI_RID_FILE*.
3. If a RID file cannot be found in these locations, the MAK RTI stops looking for the RID file and fails.

- How can I specify which FED file to use?

For MAK ONE products, use the *-F* command-line option. For most MAK ONE products, the federation name and FED file name must match. You can specify the FED file through the graphical user interface in the Logger and VR-Vantage. You can also specify a FED file in the VR-Forces Launcher.

- What happens if the RTI cannot find the RID file?

The RTI uses default values.

- When does a federation require *rtiexec*?

Some RTIs require that you always run an *rtiexec*. If you are using the MAK RTI in lightweight mode, you do not need to run *rtiexec*. If you want to use services such as synchronization points, save/restore, time management, and reliable transport you must run *rtiexec*. Please see the *MAK RTI Reference Manual* for details.

- How do I configure the RTI to use *rtiexec*?

In *rid.mtl*, set **useRtiExec** to 1. For the changes to take effect, make sure that each federate reads in a RID file that contains this change.

You can also configure use of the *rtiexec* when you add a connection in the RTI Preferences dialog box. For details, please see "Adding RTI Connection Configurations" in the *MAK RTI Users Guide*.

- How do I specify a federation to join?

For MAK ONE products, use the `-x` command-line option. For MAK ONE products, the federation name and FED file name need to match. For the vpNet module, this can be specified in LynxPrime in the NetHLAConnection.

The VR-Forces, Logger, and VR-Vantage applications let you specify the federation through their simulation connection configuration mechanisms. Please see the relevant users guides for details.

Common error messages:

- Unable to Locate Component: This application failed to start because libRTI-NG.dll was not found. Re-installing the application may fix the problem.

The HLA application cannot find the RTI libraries. Double check that an RTI is installed and that the path to the libraries is in PATH, LD_LIBRARYN32_PATH, or LD_LIBRARY_PATH.

- Error while loading shared libraries: libRTI-NG.so cannot open shared object file: no such file or directory.

The HLA application cannot find the RTI libraries. Double check that an RTI is installed and that the path to the libraries is in PATH, LD_LIBRARYN32_PATH, or LD_LIBRARY_PATH.

- ****FATAL ERROR****: Could not create Federation Execution note -> vuLibrary::Load - failed loading libvpnethla ... warn -> vpModule::initializeModule("vpNethLA") - cannot load library warn -> valid class-level tag 'NetHLAConnection' encountered, but there is no class registered.

The vpNet module cannot find the RTI libraries. Double check that an RTI is installed and that the path to the libraries is in PATH (win32), LD_LIBRARYN32_PATH (IRIX), or LD_LIBRARY_PATH (Linux/Solaris).

- ****FATAL ERROR****: Could not create Federation Execution VR-Link: RTI exception: CouldNotOpenFED ()

The HLA application could not find the FED file. Double check its location.

5.3. Using Custom FOM Configurations

When running MAK ONE products using the HLA specification with a modified FOM module list, the order of the FOM Module list can affect how the RTI processes the FOM. When the RTI processes FOM modules out of order, there can be problems.

The MAK RTI has a FOM Sorting feature that attempts to sort the submitted FOM modules based on their dependencies. This ensures that base FOM modules are processed first, so that FOM modules that rely on those base concepts are processed later. However, the means to determine the dependency hierarchy is not specified by the standard, which can make the MAK RTI FOM sorting feature inconsistent.

For this reason, we recommend that you submit FOM modules in their hierarchical order to prevent any potential issues. For example, if moduleA depends on moduleB, you should list B before A in the exercise configuration connection file or the `--setFomModuleList` command-line option. For the latter, it would be:

```
--setFomModuleList "moduleB,moduleA"
```

5.4. Using the DIS Spy

DIS Spy is a simple tool used to debug network communications in the DIS protocol. The DIS Spy visualizes packet activity in a user-friendly way. It follows the SISO Enumeration standardized document and utilizes those same enums to decode the PDU records into the user-readable format.

5.4.1 Starting the DIS Spy

► To start the DIS Spy:

1. From the Start menu, choose **MAK Technologies > product name ## Tools**. The tools folder for the product opens.
2. Use the DISSpy link to launch the product.

5.4.2 Connecting to a Simulation

By default, this tool does not use the loopback address connection. Connect using the same connection settings as the MAK ONE product you want to check.

1. In the DIS Spy GUI, choose **File > Connect**. The Connect Dialog opens.
2. Choose the appropriate IP address.
3. Enter the Port value.
4. Click OK.

5.4.3 Viewing PDU Information

Once DIS Spy connects, it listens to and picks up packets sent over the network. The packets it has picked up are listed in the Traffic by PDU Family section, sorted by PDU type.

You can quickly explore information in PDUs by using the drop-down on the toolbar, for example, change from the Network view to the PDU view and then select a type to see the timestamps, and then select a timestamp to see its details. Each of the views provides information within the desired packets.

5.5. Using Older MAK ONE Products with HLA Evolved

When using HLA Evolved:

- VR-Link 4.0, 4.0.1, and 4.0.2 are only compatible with MAK RTI 4.0, 4.0.1, 4.0.2, and 4.0.3.
- VR-Link 4.0.3 and later releases are only compatible with MAK RTI 4.0.4 and later releases.
- VR-Exchange 2.0 and MAK Data Logger 5.0 are only compatible with MAK RTI 4.0, 4.0.1, 4.0.2, and 4.0.3.
- VR-Exchange 2.0.1 and later are only compatible with MAK RTI 4.0.4 and later releases.
- VR-Vantage 1.4 and later and VR-Forces 4.0.3 and later are only compatible with MAK RTI 4.0.4 and later releases.

How to Contact Us

For MAK ONE product 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:	https://www.mak.com/mak-one/support/portal
Technical support:	https://www.mak.com/mak-one/support/portal
Product version and platform information:	https://www.mak.com/mak-one-specs
For the free, unlicensed MAK RTI:	https://www.mak.com/mak-one/support/help/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.



6. Conventions and Third-Party Licenses

This chapter explains the conventions used in this manual and MAK Technologies documentation.

Document Conventions	88
DI-Guy Conventions	88
Mouse Button Naming Conventions	89
Third-Party Licenses	89

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, (<code>--help -h</code>).
/	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 MAK ONE home directory. For example, the directory `./vrlink#.#/doc` appears in the text as `./doc`.

DI-Guy Conventions

The DI-Guy documentation uses these conventions used for entity coordinates.

Convention	Indicates
XYZ	X = forward, Y = to the left Z = up
Yaw, Roll, Pitch	Orientation is applied in the order YRP. Yaw = rz – orientation about the vertical axis Roll = rx – orientation about the fore-aft axis Pitch = ry – orientation nose down, nose up

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. For an up-to-date list of supported third-party and open source libraries and the associated license information, please visit the MAK ONE Product Support Knowledge Base (at <https://www.mak.com/knowledge>) and refer to the Third-Party/Open Source List under Frequently Asked Questions (FAQ).



Index

A

- application number, 16
- attribute record
 - DtActivityRecord, 41
 - DtAircraftLightingRecord, 43
 - DtAisDataRecord, 45
 - DtCallsignRecord, 42
 - DtDestinationRecord, 41
 - DtEmbeddedUnitListRecord, 41
 - DtGroundVehicleLightingRecord, 44
 - DtRadarBeamExtensionRecord, 46
 - DtRouteRecord, 42
 - DtStatusRecord, 40
 - DtSymbolIdRecord, 43
 - DtTankerAircraftLightingRecord, 44
 - DtUuidRecord, 40
 - DtUuidSourceUnitRecord, 40

B

- broadcast address, 80

C

- command-line option
 - exConnConfigFile, 20
 - setFomModuleList, 84
- compiler, 12
- configuration
 - file, RTI, 11
 - FLEXlm as a service, 78

- connection

- exercise configuration file, 20
 - loopback, 73

- custom PDU, See MAK object model

D

- Data Logger, 14
 - ports, 70

- database
 - effect of optimization, 22

- DDM, 11

- debugging
 - network communication (DIS), 84

- DIS
 - loopback connection, 73

- DIS application
 - incompatibility, 79
 - running, 11

- DIS Spy, 84

- dynamic terrain
 - fields, 33
 - object attributes, 33

E

- environment variable
 - RTI_CONFIG, 82
 - RTI_RID_FILE, 82
- error message, 76
 - FLEXlm, 78

F

- FDD file, 11
- FED file, 11, 76, 82
 - naming convention, 17
- federate, 11
- Flat Earth database, 21
- FLEXlm, 13
 - error messages, 78
- flexlm directory, 77
- FLEXlm Licensing ports, 73
- FLT file
 - features supported, 21
- FOM
 - file order, 84
 - module, 84

H

- HLA
 - 1516, troubleshooting, 16
- HLA 1.3, 13, 81
 - interoperability with IEEE 1516, 14
- HLA 1516, 11, 13, 81
- HLA application, 10
 - incompatibility, 81
 - startup problems, 76

I

- IEEE 1516
 - interoperability with HLA 1.3, 14
- ifconfig command, 80
- installing
 - RTI, 11
- interoperability
 - HLA 1.3 and IEEE 1516 federates, 14
- ipconfig command, 80

L

- LAN, 11, 80
- Legion
 - loopback connection, 73
 - port, 70
- library
 - RTI, 11, 76
- license management
 - troubleshooting, 77

- license server, 10, 77
 - configuring as service, 78
- lightweight mode, 11
- linking to VR-Link and RTI, 16
- Linux, 12, 76
- log file, 76
- Logger, 14
 - interoperability of HLA 1.3 and IEEE 1516, 14
 - ports, 70
 - recording to from VR-Forces, 16
- loopback connection, 73

M

- maintenance agreement, 78
- MAK Data Logger, 14
 - object model, 39
 - ports, 70
- MAK Legion
 - loopback connection, 73
 - port, 70
- MAK object mode
 - NETN METOC, differences, 51
- MAK object model, 23-24
 - date, 49
 - extension
 - aerodrome, 29
 - aggregate, 33
 - DI-Guy, 38
 - dynamic terrain, 33, 39
 - human, 38
 - Logger, 39
 - physical entities, 39
 - runway, 29
 - terrain, dynamic, 33, 39
 - extensions, 29
 - FOV, 47
 - radar, 46
 - sensor field of view, 47
 - time, 49
 - weather, 51
- MAK RTI, 10, 13
 - compatibility with old applications, 85
 - error messages, 76
- MAK RTI port, 70
- maklmgrd executable, 10
- MAKLMGRD_LICENSE_FILE environment variable, 77
- METOC, 51
- Microsoft Visual C++ 6.0, 12
- MOM, 11

N

NETN METOC
 MAK, differences, 51

O

object model, MAK, See MAK object model
 Open Flight database, 21
 operating system
 compatibility, 12

-

-P command-line option, 11

P

path, 76
 PDU
 aerodrome, 29
 dynamic terrain, 35
 dynamic terrain version, 33
 runway, 30
 sensor field of view, 47
 time and date, 50
 visualizing, 84
 ping, 77
 port
 3000, 11
 DIS, 11
 FLEXlm Licensing, 73
 Legion, 70
 Logger, 70
 MAK Data Logger, 70
 MAK Legion, 70
 MAK RTI, 70
 RadarFX, 72
 RTI, MAK, 70
 Tracy, 73
 VR-Engage, 72
 VR-Exchange, 71
 VR-Forces, 71
 VR-Link, 70
 VR-Vantage, 71
 WebLVC, 71

Q

Qt, 12

R

RadarFX ports, 72
 recording
 to Logger, 16
 requirements
 basic application, 10
 RID file, 82
 search order, 82
 rid.mtl, 11
 RTI
 configuration file, 11
 configuration settings, 82
 installation, 11
 libraries, 76
 linking to, 16
 version, 81
 RTI library, 11
 RTI port, MAK, 70
 RTI.rid, 11
 RTI_CONFIG, 76
 RTI_CONFIG environment variable, 81-82
 RTI_RID_FILE environment variable, 82
 rtiexec, 11, 76

S

search order
 RID file, 82
 startup problem, 76
 Stealth, 14
 synchronizing terrain with VR-Forces, 21
 Stealth Control PDU, 16

T

terrain
 synchronizing between Stealth and VR-Forces, 21
 third party API, 12
 Time Management, 11
 Tracy port, 73
 troubleshooting, 76

U

UDP port, 11
udpPort, 81
useRtiExec, 83
UTM, 21

V

variable, environment, See environment variable
viewing
 Logger files, 16
vpNetHLA, 76
VR-Engage port, 72
VR-Exchange ports, 71
VR-Forces, 13, 16
 recording to Logger, 16
 synchronizing terrain with Stealth, 21
VR-Forces ports, 71
VR-Link, 13
 linking to, 16
VR-Link port, 70
VR-Vantage ports, 71

W

WebLVC ports, 71
WGS84 datum, 21

