VR-Link 5.5 Release Notes

This release note provides the following release-specific information for VR-Link Release 5.5:

Document ID: VRL-5.5-3-200610

Systems Supported

Table 1 lists the platforms currently supported by MAK VR-Link 5.5. Please contact MAK if you are interested in purchasing VR-Link for other platforms. Application code must be built with the indicated compilers in order to link to VR-Link libraries.

Table 1: Platforms supported

Platform	Compiler
Red Hat Enterprise Linux Workstation 7.0 (64-bit libraries)	default compiler
Red Hat Enterprise Linux Workstation 8.0 (64-bit libraries)	default compiler
PC with Windows 10	Microsoft Visual C++ 12.0 (64-bit) Microsoft Visual C++ 14.0 (64-bit)

The VC 14 build is binary compatible with all MAK VC 15 products.

VR-Link for C# supports all Windows compilers.

VR-Link for Java supports all compilers and operating systems.

VR-Link for Java requires Java 8.

Disk Space Requirements

Linux installations require approximately 2 GB of disk space. On Windows, you need approximately 1.5 GB of disk space.

Compiler Compatibility on Windows

MÄK provides versions of product releases that have been compiled with different versions of Microsoft Visual C++. When you run MÄK products together on the same computer, for example, the Logger and a VR-Vantage application, we strongly recommend that you run versions compiled with the same compiler. Mixing products compiled with different versions of the compiler on the same computer can result in program instability. The exception to this is that products built with VC 14 and VC 15 are binary compatible.

License Manager

To run the licensed version of VR-Link, you must install license management software. VR-Link 5.5 uses FLEXlm 11.13.0.2. If you are upgrading from a version of the VR-Link that used an older version of FLEXlm, you must upgrade your license management files. You do not need a new license. Licenses are forward compatible.

The License Manager files are not part of the VR-Link installer. You can download them at:

- Windows:
  http://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (for 32-bit systems)
  http://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- Linux: http://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz (for 32-bit systems)
  http://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

License manager support is available at:

http://www.mak.com/support/licenses

Network Compatibility

VR-Link 5.5 is compliant with RPR FOM 1.0 and RPR FOM 2.0 and includes FOM mappers for many draft versions of the RPR FOM. It is compliant with RTIs that support the HLA 1.3 specification, the SISO DLC HLA API 1516 version of the IEEE 1516 specification (SISO-STD-004.1-2004), or HLA Evolved.

To use an RTI with VR-Link, it must use the same operating system and be built with the same compiler.

VR-Link 5.5 supports DIS 4, 5, 6, and 7, and can therefore interoperate with DIS applications of any of these versions.

Backward Compatibility

VR-Link 5.5 has the following backwards compatibility issues:

- VR-Link versions 5.0-5.1.3 encoded environmental process DIS Version 7 PDUs incorrectly. This has been corrected. Unfortunately this means that VR-Link 5.3.1 will not inter-operate with those versions of VR-Link for DIS 7 Environmental Process PDUs. We recommend upgrading all VR-Link 5.0-5.1.3 federates if you are using that specific configuration.
- VR-Link 5.2 encoded data interactions incorrectly for RPR FOM 2.0. This is fixed in VR-Link 5.3.1. However, this incompatibility causes applications that were built with VR-Link 5.2 to crash when trying to decode VR-Link 5.3.1 interactions.

  For backwards compatibility, the incorrect encoding is now provided as RPR 2.0 Revision 1. The correct encoding is RPR 2.0 Revision 2. Use Revision 1 to maintain compatibility with previous versions of VR-Link. Use Revision 2 to be compliant with RPR FOM 2.0.
- VR-Link 5.0 renamed header files to conform to MAK file naming conventions. If you have not already migrated to these new file names, the next two sections describe the changes and the migration script.

Using the Compatibility Directory

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

Using the Migration Script

VR-Link provides a script (*./compatibility/updateIncludes.sh* on Linux, *./compatibility/updateIncludes.exe* on Windows) that will go through all of your source files and modify the include statements to reference the correct header files.

The update script uses the mapping files located in *./compatibility/mappings* to update include statements for the renamed VR-Link header files in a given set of source directories. The script searches all input directories for header (*.h and *.hpp), inline (*.inl), and source files (*.c, *.cpp, and *.cxx), and checks the include statements in those files against the mapping files. For the script to correctly map the header file names, VR-Link include statements must specify the VR-Link library path. This is so the scripts can identify which mapping file to use to map the include statement. Therefore, the base VR-Link include directory must be set as the include path for projects, instead of adding the individual library directories (*include/vl*, *include/mtl*, and so on.).

The script does the following:

- If a header file has been renamed, the header file in the include statement is renamed.
- If a header file has been removed, the include statement is removed.

The command syntax for the script is as follows:

```
updateIncludes [options] directory1 ... directoryn
```

where:

- `directory1 ... directoryn` is a list of directories in which the script will search for source and header files for updating.
- `-d` displays debug level information about the files and include statements as they are processed
- `-h` provides information about the various flags.
- `-v` displays slightly more verbose output than `-d`.
- `-x` tells the script to perform the replacements. If the script runs without the `-x` flag, it outputs the commands that will be run if the `-x` flag is specified. This is so you can verify the changes that are going to be made before the script changes all the files.

RTI Support

VR-Link 5.5 has been tested with the MAK RTI 4.5.1 for HLA 1.3, HLA 1516, and HLA Evolved.

If you use the MAK RTI, remember to make sure that VR-Link can find your FED file or FDD file, and optional *rid.mtl*. Put the FED file or FDD file in the directory from which you are running, or set the environment variable RTI_-CONFIG to the directory that contains it. See *MAK RTI Reference Manual* for a list of the options for specifying the location of the *rid.mtl* file.

Applications built with VR-Link 5.5 for HLA can use any RTI that conforms to the relevant dynamic link compatible HLA standard (1.3, SISO DLC 1516, HLA Evolved) and was built using the same operating system and compiler.

New Features and Changes

VR-Link 5.5 has the following new features:

- The received timestamp was added to C# and Java objects and interactions. The timestamp is either the received time (for relative timestamps) or the value of the timestamp field (for absolute timestamps). The timestamp comes from the received DIS PDU or from the HLA tag. It matches the VR-Link C++ member method lastReceivedTimeStamp().
- DI-Guy entity state data can now be sent as attributes in attribute PDU (coupled or not) as well as the previous approach of sending separate custom DI-Guy PDU.
- Support has been added for new DIGuy Attribute "DIGuyPatches". This attribute contains information for Rank, Unit, Personal Unit, and Name patches for DIGuy character uniforms. To support this new attribute, the default RPR FOMs were updated and a new DIS attribute record was added.

Fixed Bugs

This release fixes the following issues that were problems in previous releases:

- PDUs with coupled Attribute PDUs no longer grow in size unnecessarily. VRL-206
- XML-evolved filer writer now writes parts of variant types as elements instead of attributes. VRL-212
- Position and orientation of dead reckoning has been fixed in Java libraries. VRL-214
- C# applications now handle federateName command line argument correctly. VRL-215
- C# generated code no longer uses reserved keywords. VRL-217
- C# generated code no longer uses bad characters in generated identifiers. VRL-218
- C++ code generator now correctly encodes complex data types in HLA13. VRL-219
- makVrfVrlinkExtToolkit was overriding the DI-Guy PDUs and objects. Upgraded DI-Guy objects to resolve this issue. VRL-220
- Made improvements to the DIS Entity and Aggregate Publishers to limit the amount of Relative Location PDUs sent for embarked entities. These PDUs are now sent only when relative attributes change or on a set timeout threshold. Default threshold is 5 seconds. 68274
- The DtEntityStateRepository class function "setDiGuyAction(const DtString& val, bool paused, double speed = 0.0)" has been replaced with setDiGuyAction(const DtString& val, bool paused, double speed = 0.0, bool useActionPositioning = false, const DtVector& offset = DtVector(0, 0, 0)). 68311
- Actions using action positioning and offset were not receiving that data. 68322
- Specifying a federate name for HLA Evolved caused base FOM module to be excluded from joinFederationExecution service. VRL-231
- DI-Guy hand items were read incorrectly. VRL-226
- Java in-place decoding was not working. VRL-213
- isBroadcastAddrString failed if subnet was smaller than expected for IP address class. VRL-208
- The ActiveSonarBeamEncoder/Decoder did not register attributes for rprFomVersion 2.0. VRL-204
- Fixes issue with passing a string argument on the command line with a space character. If adding a variable for the command line that might contain spaces, use std::string type. VRL-203
- IFF missing attribute en/decoders for HLA. VRL-188
- Crash when reflected designator times out. VRL-175
- Could not change rprFomRevision in Java or C# API. VRL-167