VR-Link 5.3 Release Notes

This release note provides the following release-specific information for VR-Link Release 5.3:

Document ID: VRL-5.3-3-161206

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 5.3. Please contact MÄK 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 5.0, 6.0 (32 bit and 64 bit libraries)	default compiler
Red Hat Enterprise Linux Workstation 7.0 (64 bit libraries)	default compiler
PC with Windows Vista/Windows 7/ Windows 8, Windows 10	Microsoft Visual C++ 10.0 (32-bit and 64 bit)
	Microsoft Visual C++ 12.0 (64 bit)
	Microsoft Visual C++ 14.0 (64 bit)

VR-Link for C# supports all compilers and operating systems.

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, 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.

License Manager

To run VR-Link, you must install license management software. All versions use FLEXlm 11.13.0.2. If you are upgrading from a version of 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:
  ftp://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (for 32-bit systems)
  ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- Linux:
  ftp://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz (for 32-bit systems)
  ftp://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

A license manager FAQ is available at:

http://www.mak.com/license-support.html

Network Compatibility

HLA only

VR-Link 5.3 is compliant with:

- RPR-FOM 0.5, 0.7, 0.8, 1.0, RPR FOM 2.0 drafts 6, 14, 17, and 20, and RPR FOM 2.0.
- MÄK RTI 2.x, 3.x, 4.x
- Pitch RTI HLA Evolved C++ interface.
- Other 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.

DIS only

VR-Link 5.3 supports DIS 4, 5, 6, and 7 and can therefore interoperate with DIS applications of any of these versions.

Backward Compatibility

VR-Link 5.3 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 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. However, this incompatibility causes applications that were built with VR-Link 5.2 to crash when trying to decode VR-Link 5.3 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 MÄK 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.

RPR FOM Versions Supported

VR-Link 5.3 has built-in support for

- RPR FOM 0.5, 0.7, 0.8, 1.0
- RPR FOM 2.0, drafts 6, 14, 17, and 20
- RPR FOM 2.

By default, all VR-Link examples use RPR FOM 2.0. The *DtVrlApplicationInitializer* class, however, still defaults to using RPR FOM 1.0.

VR-Link does not officially support RPR FOM 2 Draft 18. However, the draft appears to be similar enough to RPR FOM 2 Draft 17 that the 2.0017 FOM Mapper may be used for federates wishing to interoperate with RPR FOM 2 Draft 18.

If you want to use a version of the RPR FOM other than 1.0, pass the version number (0.5, 0.7, 0.8, 2.0006, 2.0014, 2.0017, 2.0020) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExerciseConn* constructor. Also, make sure you are using a federation execution name that corresponds to the right FED file. For example:

```
DtExerciseConn conn("MAK-RPR20017-1-1.fed", "MyApp", new
  DtRprFomMapper(2.0017-1));
```

VR-Link examples like *f18* and *hlaNetdump* have a command line option, `--rprFomVersion`, that you can use to choose a RPR FOM version, using one of the version numbers listed in the previous paragraph.

RPR FOM 2.0 and HLA Evolved

In previous versions of VR-Link, custom extensions to the RPR FOM were included in the HLA Evolved XML file. For VR-Link 5.3, we provide an XML file that only includes the standard RPR FOM 2.0 and we use FOM modules for VR-Link extensions. This should make it easier for customers who do not need the FOM extensions to interoperate with non VR-Link federates. The files supplied are:

- *RPR_FOM_v2.0_1516-2010.xml*
- *MAK-DIGuy-1_evolved.xml*
- *MAK-LgrControl-1_evolved.xml*
- *MAK-VRFExt-1_evolved.xml*
- *MAK-VRFAggregate-1_evolved.xml.*

RTI Support

VR-Link 5.3 has been tested with the MÄK RTI 4.4.1 for HLA 1.3, HLA 1516, and HLA Evolved. It has also been tested with Pitch's pRTI 4.3.1 for HLA 1.3 and HLA Evolved.

If you use the MÄK 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 *MÄK RTI Reference Manual* for a list of the options for specifying the location of the *rid.mtl* file.

Applications built with VR-Link 5.3 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 5.3 has the following new features:

- Added support for ownership transfer in DIS, using the DIS 7 standard. The interface for transferring entities is similar to the existing ownership transfer in HLA, and the objectOwnership example has been updated to support both protocols at the same time.
- You can now set the precision of smoothing using the *DtExponentialSmoother* (or *DtOldSmoother*) by calling the following function:

  ```
  void DtSmoothOrient::setDfltSmoothPrecision(double t);
  ```

  By default it is set to 10e-12.
- There are now new factories for encoders and decoders in DIS. For details, please see "New DIS Factories for Encoders and Decoders," on page 8.
- The minidumper now supports headless applications.
- Support for conversion functions for big and little endian functions in C#.

New DIS Factories for Encoders and Decoders

There are new factories for encoders and decoders in DIS in addition to HLA for anyone who wants to change how DIS encoding is done. You can set these new classes like this:

```
exConn()->pduEncoderFactory()->addEncoder(DtPduTypeToSet,
  new DtSpecialEncoder);
exConn()->pduDecoderFactory()->addDecoder(DtPduTypeToSet,
  new DtSpecialDecoder);
```

Additionally, we have added a way to change the state repository of DI-Guy information by doing the following.

```
DtDIGuyObjectPublisher::setStateRepCreator(
  (DtDIGuyObjectPublisher::DtStateRepCreator)
  DtSpecialDIGuyObjectStateRepository::create);
DtReflectedDIGuyObject::setStateRepCreator(
  (DtReflectedDIGuyObject::DtStateRepCreator)
  DtSpecialDIGuyObjectStateRepository::create);
```

The same call works for both HLA and DIS.

We have also added new decoder and encoder classes to DI-Guy callbacks in DIS allowing you to customize it just like you can customize the HLA side. Note that if you want to add new extensions to these encoders and decoders, these classes have callbacks for you to do that - it is not necessary to create a new derived class.

Fixed Bugs

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

- Resolved issue which joinMulticast could fail when specifying a device other than 0.0.0.0 in Linux. 53381
- Fixed memory leak on entity state instantiation. 56815
- Suppress Self-Reflect specific set methods for socket options were removed and the behavior of the primary set methods were modified to perform the necessary logic depending on whether Suppress Self-Reflect is enabled or not. 56821
- Fixed crash bug in printDataToStream method for generic state repositories. 57147
- Fixed crash bug in printDataToStream method for generic state repositories. 57148
- If myMomIsDisabled was not initialized, it behaved differently in debug and release. 57193
- If asynchIO was enabled in the MTL file and `--useAsyncIO` was entered on the command line, asynch IO got turned off. 57194
- Burst descriptors in fire and detonate interactions now include accessors and support all three possible types of descriptors: Munition, Explosion, or Expendable, and what they are is determined by the kind of the EntityType defined in this burst descriptor. Note that some accessors are only relevant when using the right kind of descriptor. if you, for example, ask for ExplosiveForce in a Munition descriptor what is returned is an undefined value. 57969

- DI-Guy objects were not extendable. 58196
- Fixed bug in decode logic for SEES Vectored Nozzle Systems in HLA that caused update spam and undecoded values. 58680 58681
- In previous versions of VR-Link, RPR 2.0 Variable Datum Sets were sent using the RPR 2d14 encoding, which was incorrect. VR-Link 5.3 fixes the encoding to be correct for RPR 2.0, but this breaks compatibility with software built on an older VR-Link, which is still expecting the 2d14 encoding. Please see "Backward Compatibility," on page 4 for more information. 59109

Known Problems

This release of VR-Link has the following known problems:

- The default configuration for all VR-Link for C# examples is 64-bit debug. If your target is 32-bit or release, you must change the configuration.
- The VR-Link Code Generator may have problems with complex FOMs. We have made every effort to verify that it works with a broad range of FOMS and FED files. However, we are unable to verify that it works for all FOMs. In addition, small errors in the FOM may prevent the Code Generator from correctly processing the FOM at all. If the Code Generator is not able to process your FOM, we would be happy to help. Please send a copy of your FOM to support@mak.com and we will work with you to correctly generate your FOM.
- VR-Link correctly encodes and decodes EnvironmentalProcess objects for RPR FOM 2, Draft 14. However, the RPR FOM specifies a way of encoding which prevents VR-Link from correctly decoding modified or custom EnvironmentalRecords. Unfortunately, many MÄK products built with VR-Link use customized EnvironmentalRecords. If you use these products or if you use customized records in your applications, please do not use the RPR FOM 2 Draft 14 FOM Mapper. This is not a problem in other FOM versions, and it has been resolved in future versions of the FOM.
- The RTI 1516 API specifies that all strings passed to and from the RTI are handled as wide strings. VR-Link (and the MÄK RTI) store strings internally as narrow strings. This means that true multibyte wide strings used by RTI Federates may not be handled correctly when received by VR-Link. VR-Link handles name reservation wide strings correctly, however, as this is a 1516-only function. VR-Link provides conversion functions from Wide To Narrow/Narrow to Wide string conversion in *vlStringUtil.h*.
- On non-windows platforms, the classes *DtSharedMemoryPoolManager* and *DtSharedMemoryPoolClient* (*vlShmPool.h*) sometimes have difficulty creating a large memory pool (greater than 1 MB) if the pool name is longer than two or three characters. The classes refuse to create the memory pool even if the requested pool is smaller than the system resource SHMMAX. Setting the pool name to fewer than three characters will work around this problem.