VR-Link 5.1.3 Release Notes

This release note provides the following release-specific information for VR-Link Release 5.1.3:

- Systems Supported 2
  - Disk Space Requirements 2
  - Compiler Compatibility on Windows 2
  - License Manager 3
- Network Compatibility 3
- Backward Compatibility 4
  - Using the Compatibility Directory 4
  - Using the Migration Script 4
- RPR FOM Versions Supported 5
- RTI Support 6
- New Features and Changes 6
- Fixed Bugs 7
- Known Problems 7

Document ID: VRL-5.1.3-3-150512

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 5.1.3. 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	Microsoft Visual C++ 8.0 Microsoft Visual C++ 10.0 (32-bit and 64 bit) Microsoft Visual C++ 11.0 (32-bit and 64 bit)

VR-Link for C# supports VR-Link 5.1.3, Microsoft Visual C++ 10.0 and MSVC++ 11.0, 32 bit and 64 bit.

Disk Space Requirements

Linux installations require approximately 1.2 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. The VC++ 8.0 version of VR-Link 5.1.3 uses FLEXlm 11.11. All other 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 5.1.3 is compliant with:

- RPR-FOM 0.5, 0.7, 0.8, 1.0, and a subset of 2.0 (draft 6, 14, 17, and 20)
- MÄK RTI 2.x, 3.x, 4.x (HLA 1.3 and 1516-2000 only)
- MÄK RTI 4.0.4 or later (HLA Evolved)
- Pitch RTI 1.3 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), and 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 5.1.3 supports DIS 4, 5, 6, and 7 and can therefore interoperate with DIS applications of any of these versions.

Backward Compatibility

Effective with VR-Link 5.1, *DtReflectedObject* lists no longer use the deprecated class *DtObjectIdHashList*. Instead they use std::map which is much faster. Unfortunately this does cause a slight change to all *DtReflectedObject*. Previously they were *DtListedObj*, and now they are not. This should not affect your VR-Link code unless you have created new HLA classes (for a FOM Mapper for example).

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 5.1.3 has built-in support for versions 0.5, 0.7, 0.8, 1.0, and 2.0, drafts 6, 14, 17, and 20, of the RPR FOM. By default, VR-Link 5.1.3 uses 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("VR-Link20017", "MyApp", new DtRprFomMapper(2.0017));
```

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 Draft 20 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 5.1.3, we provide an XML file that only includes the standard RPR FOM 2.0 draft 20 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_draft20_1516-2010.xml*
- *DIGuyModule.xml*
- *VR-Link_LgrControl_evolved*.xml
- *VR-Link_ViewControl_evolved*.xml
- *VR-Link_VrlExt_evolved*.xml.

RTI Support

VR-Link 5.1.3 has been tested with the MÄK RTI 4.2 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 the 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 5.1.3 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 5.1.3 is primarily a maintenance release. It has the following new features:

- Added support for Red Hat Enterprise Linux Workstation 7.0.
- Dropped support for Red Hat Enterprise Linux Workstation 4.0, SUSE 11, and MS VC++ 7.1 and 9.0.
- The Code Generator now uses VR-Link structures when it can (for example, *DtVector* and *DtString*).
- There is a new example "TalkEvolved" and "ListenEvolved" that shows how to use some of the HLA Evolved (1516-2010) specific features.
- For all platforms except MSVC++ 8, the Threading Building Blocks library has been updated to version 4.3.4. VC 8 continues to use version 4.0.

Fixed Bugs

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

- Compiling 32bit and 64bit examples together now works properly. 49997
- The documentation now has a better description of when to use different dead reckoning algorithms instead of just listing all the options VR-Link provides. 52418
- Under some circumstances it was possible the DtReflectedObjectList::lookupObjInWaitList would go in an infinite loop. 54079
- The DIGuyParent field inside a Physical entity will now correctly encode a nullEntityIdentifier instead of a null attribute if there is no available parent. 54447
- In C#, the SiteId (-s or --siteId) now works. 54484
- If an entity is in the center of the earth, it will no longer start smoothing until it receives a valid location. 54647
- The *DtSmoother* and the *DtOldSmoother* classes now correctly stop smoothing when an entity's frozen bit is turned on. 54951
- The code generated examples now load the correct FOM as well as FOM Modules instead of VR-Link. 48096
- Generated VR-Link code now looks a little nicer, as we have cleaned up a lot of the whitespace inconsistencies in each generated file. 49237
- Removing items from the code generator menu now correctly stops generating those items. 54457
- In the code generator, it is now possible to generate an HLA Evolved project without including HLAStandardMIM.xml. 54459
- Saving an empty project in the code generator will no longer stop you from importing a FOM later. 53979

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.
- Section 5.10, “Interoperability Between HLA 1.3 and IEEE 1516 Federates”, in *VR-Link Developers Guide* describes interoperability issues that arise when you try to run an HLA 1.3 federate with an IEEE 1516 federate. In particular it describes the requirement to use an identical FED or FDD file for all federates. In future releases, we intend to rectify this situation, and allow run-time interoperability between federates that have loaded a 1.3-style FED file and federates that have loaded a 1516-style XML file that otherwise includes the same set of classes, attributes and parameters.