VR-Link 5.2 Release Notes

This release note provides the following release-specific information for VR-Link Release 5.2:

Document ID: VRL-5.2-3-160105

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 5.2. 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) Microsoft Visual C++ 12.0 (64 bit)

VR-Link for C# supports:

- VR-Link 5.2.
- Microsoft Visual C++ 10.0 and MSVC++ 11.0 32 bit and 64 bit.
- MSVC++ 12.0 64 bit.

VR-Link for Java supports:

- VR-Link 5.2.
- Microsoft Visual C++ 10.0 and MSVC++ 11.0 32 bit and 64 bit.
- MSVC++ 12.0 64 bit.
- Red Hat Enterprise Linux Workstation 5.0 and 6.0 32 bit and 64 bit.
- Red Hat Enterprise Linux Workstation 7.0 64 bit.

VR-Link for Java requires Java 8.

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 5.2 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 5.2 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 5.2 supports DIS 4, 5, 6, and 7 and can therefore interoperate with DIS applications of any of these versions.

Backward Compatibility

VR-Link versions 5.0-5.1.3 encoded environmental process DIS Version 7 PDUs incorrectly. This has been rectified for this release of VR-Link. Unfortunately this means that VR-Link 5.2 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 5.0 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 5.2 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, VR-Link 5.2 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("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 5.2, 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 5.2 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 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 5.2 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 5.2 has the following new features:

- FED files and XML files have been updated and naming has been standardized. For details please see "New FED File Naming Convention," on page 8.
- Support for Java. VR-Link for Java consists of both Java code and platform-specific code. Because of this dependency on native code, VR-Link for Java is not portable to any Java platform; only platforms supported by VR-Link (C/C++). Both 32- and 64-bit native libraries are provided for the platform corresponding to the VR-Link (C/C++) installation. For complete details, please see *VR-Link Developers Guide*.
- The Minidumper now prints source file and line number in the stack trace when debug symbols are available. This will not provide files and line numbers for most shipped libraries, but will provide information for customer code or examples if rebuilt with debug symbols.

- Added timestamping and log file rolling support to DtFilePrinter. The method signature for *DtPrinter::write* (and all derived classes) has changed - new argument `bool sync`.
- The F18 examples now includes a reflected emitter beam list to show a user how you can use VR-Link to reflect information as well as publish.
- When using printDataToStream(), some attributes, such as location and velocity, used to define their own precision instead of using the stream's (setprecision) precision values. This caused a limit on the maximum amount of precision you could ever display. This has changed. Now VR-Link always uses the stream's precision values. However, that the default precision is smaller than the precision we were originally setting, so by default some printouts may provide fewer digits.
- The Point Object state repository is now implemented the same way as Areal and Linear state repositories.
- Areal and Linear Environment Objects are now supported in DIS.
- Talk, Listen, and Netdump examples now use the VR-Link shutdown callback to shut down cleanly when forced by the MAK RTI.
- Multiple improvements to the Code Generator:
  - All code generated fixed array types now have two set functions instead of one. The second set function allows you to set the entire array at once, instead of going object by object:

```
virtual void setArrayItem(int index, DtU8& val);
virtual void setArray(const DtU8 val[size]);
```

  - The Code Generator will now only generate datatypes of the classes that you are specifically using. It previously generated all datatypes at all times, creating a more complex and complicated project than was necessary.
  - The Code Generator now supports command line arguments. For a list of arguments, enter:

```
./bin/vrlCodeGenerator -h
```

  - The Code Generator can now distinguish between object and interaction classes of the same name.
  - The Code Generator now creates a solution file as well as project files so that you can compile all the examples and the main project at once.
  - The Code generator now adds a number (1, 2, 3, and so on) to the end of the name of an object that has the same name as a different object.

    For example, HLAFederation/setSwitches generates to DtSetSwitches. HLAFederate/setSwitches would now generate to DtSetSwitches1.

- There are two new command line arguments. These arguments are available to any VR-Link application that uses the *DtExerciseConnInitializer*, such as talk, listen, and f18:
  - `--logFileName` *`filename`*. VR-Link print messages display to both the console and this file.
  - `--fileNotifyLevel` *`level`*. Allows you to log to a file at a different notify level than logging to the console. If this is not set, it defaults to the value set by `--notifyLevel`.

New FED File Naming Convention

Historically, VR-Link has 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 has two problems. First, it is 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 this release, VR-Link introduces 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* has been renamed *MAK-RPR20017-1-1.fed*.

Additionally, given the power of HLA Evolved FOM Modules, starting with RPR 2.0 we are now 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 will automatically add 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. Over the coming year, as other VT MÄK application move up to this release of VR-Link, they will begin to use the FED files with the new naming conventions. The old FED files will then be deprecated.

Table 2 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 2: Old FED file to new FED file mappings

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

Table 2: Old FED file to new FED file mappings

Old FOM	New FOM
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	
VRL-DIGuy20017DDM.xml	

Table 2: Old FED file to new FED file mappings

Old FOM	New FOM
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_with-UpdateRate.xml	
VR-Link20017-1_evolved.xml	MAK-RPR20017-1-1_evolved.xml
VR-Link20017-1_evolved_withUpdat-eRate.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

API Changes

This section lists the major changes to the API, with porting instructions.

- *DtVector* has been split into *DtVector32* and *DtVector64*. (*DtVector64* is aliased to *DtVector*.) All functions that take velocity, acceleration, and embedded position information have been converted into *DtVector32*. This was done because DIS and the RPR FOM use 32-bit values for those fields and customers using our standard *DtVector* were seeing small rounding issues due to conversion to and from 32-bits when the data was passed over the network. Unfortunately, this also affects a large number of classes in VR-Link.

  For convenience, we have added DtVector32To64 and DtVector64To32 conversion functions, but we highly recommend that you avoid converting between 64 and 32 bits as much as you can.

- *DtOldSmoother* has been renamed *DtExponentialSmoother*. *DtSmoother* has been renamed *DtTrigonometricSmoother*. This better reflects the fact that both smoothers are legitimate ways to smooth, rather than there being a new way and an old way. In fact, we currently recommend you use the *DtExponentialSmoother* by default, as we have found that provides better smoothing for most cases. Because of this, the new default smoother is *DtExponentialSmoother*.

  The header files still provide typedefs for the *DtOldSmoother* and *DtSmoother*, and we have placed the old header files in the compatibility directory so that old code will still work.

- New addObjectClassName and removeObjectClassName functions now allow users to easily decide which specific HLA classes are being handled by your reflected objects. For example, the following code removes all Physical entity subscriptions except for aircraft:

```
#ifdef DtHLA
      DtFomMapper* rprFom = exConn.fomMapper();
      rprFom->removeObjectClassName("DtReflectedEntityList",
         "HLAobjectRoot.BaseEntity.PhysicalEntity",true);
      rprFom->addObjectClassName("DtReflectedEntityList",
         "HLAobjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft",
  false);
#endif
```

  For a number of performance reasons related to string comparisons, the *DtFomMapper* classes related to this have now been replaced from *DtList* and *DtHashtable* to std::maps and std::set. *DtFomMapper* extensions using these objects will need to be updated.

  Also, it was found that there were a number of places where VR-Link was returning `char *` values when it really should have been returning `const char *` values. These have been fixed, and may require a minor code change if you were relying on these values.

- DtInteraction addCallback functions have also replaced DtList with an std::set in order to match the DtFomMapper.

- Customers extending DtPrinter must change the write function from:

  ```
  write( const std::string &str, int notify_level)
  ```

  to:

  ```
  write( const std::string& str, int notify_level, bool newLine )
  ```

  If you want to keep the old functionality you can ignore the newLine parameter. But this new parameter can be useful to format output, for example, adding time-stamps to all messages.

Fixed Bugs

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

- VR-Link's C# API now provides the static method ExerciseConnection.Check-VRLinkLicense() to check licenses. 56066
- All the C# examples now have “Enable unmanaged code” set, so that it is possible to debug them through visual studio. 55945
- When running with the MAK RTI using lightweight mode, VR-Link examples printed an excessive number of notification messages. 55845
- On an HLA attribute request, VR-Link will now only send the attributes that were set when using generic encoding, instead of also sending even the attributes that were never set. 55939
- 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. 56792
- Fixed memory leak on entity state instantiation. 56813
- The ownership management classes (*DtDefaultOwnershipHandler* and *DtPartialOwnershipHandler*) have replaced the callback functions set(Partial)AcquireOwnershipCallback and set(Partial)DivestOwnershipCallback with add and remove methods. This was done in order to allow for multiple callbacks for each ownership change. Customers using the old set method should just replace set with add and should see no difference in functionality. 55894
- In DIS, the detonation PDU now prints its articulated part records when calling printDataToString or printDataToStream. 55056
- The F18 example no longer moves the Azimuth Center to simulate emitter beam movement. This was contrary to the way emitter systems are defined in DIS and RPR. Instead, the frequency of the beam is set and the beam remains static. 51924
- VR-Link now statically links *libxml2* on Windows platforms. Related DLLs that were previously located in the *bin* directories will no longer be present, and customers that need these can obtain them from their source. http://xmlsoft.org/ 54588
- DI-Guy Action Speed is now correctly encoded in both DIS and HLA. Previously it was only encoded in HLA. 53921

- There is a new option in the HLA *DtExerciseConnection* called passiveMode. When you set passive mode, all HLA subscriptions are passive and no requests are made in the HLA Network. This means your federate will listen to the HLA traffic but will not generate new traffic. If a passive federate is the only thing subscribed to an object, then no updates will be generated by that object.

  In order to activate this mode you can call setPassiveMode in the exercise connection before any subscriptions or reflectedObjectLists are created. 52604
- It is now easier to remove certain types of objects from the entityList. 55238
- The netutilities installer now includes 64 bit executables. 55275
- VR-Link will now check for *.xml* and *.fed* if it is given a FED file with no extension. 12208
- DtGlobalObjectDesignator now has a cross-protocol method that returns a null global ID (nullId). 12397
- The Environmental Process Publisher now sets environmental status to 0 before sending a final PDU, as defined by the DIS standard. 39742
- syntheticEnvironmentalObjectType has been moved from the *vl* folder to the *vlpi* folder to correctly identify that this is a protocol independent class. The old include file has been moved to the compatibility folder. 51026
- DataType files generated by the VR-Link Code Generator no longer have Dt in front of the file name, thus matching the naming convention in the rest of VR-Link. 54081
- In some circumstances, the Code Generator determined that the octetBoundary was 0 when it should be something else. This has been fixed and object boundaries are now calculated at runtime. 55316
- The Code Generator defined classes out of order. 48526

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.