



MÄK RTI 4.0.1 Release Notes

This document provides the following release-specific information for MÄK RTI 4.0.1:

Supported Systems	2
Using Libraries and Binaries Built with Visual Studio 2005 and Later	2
Compiler Compatibility on Windows	3
License Manager	3
Third Party Library Dependencies	3
Patch Required for AMD Dual-processor Windows PCs	4
Backwards Compatibility	4
Network Compatibility	4
Compatibility with Other RTIs	5
New Features and Updates	6
Bug Fixes	6
Known Problems	7
RTIs Spy Diagnostic GUI Scalability	8
HLA Evolved API Issues	9

The MÄK RTI has been verified by DMSO as fully compliant with the IEEE 1516 version of the HLA Interface Specification (SISO DLC HLA API 1516 (SISO-STD-004.1-2004).)

Copyright © 2010 VT MÄK, 68 Moulton St., Cambridge, MA 02138 All rights reserved.
MÄK Technologies®, VR-Forces®, B-HAVE®, RTIs Spy®, and VR-Link® are registered trademarks of VT MÄK. Document ID: RTI-4.0.1-2-101020

Supported Systems

Table 1 lists the platforms supported by the standard version of MÄK RTI 4.0.1. Please contact MÄK if you need it for another platform. Application code must be built with the indicated compilers in order to link to the MÄK RTI libraries.

Table 1: MÄK RTI: Platforms supported

Operating System	Compiler
Red Hat Enterprise Linux Workstation 4	Default compiler.
Red Hat Enterprise Linux Workstation 5. (32 bit and 64 bit libraries)	Default compiler.
Windows XP	Microsoft Visual C + + 7.1, 8.0, 9.0**
Windows Vista*/Windows 7	Microsoft Visual C + + 8.0, 9.0**
*You must have administrator privileges to install VT MÄK products on Windows Vista.	
**The MSVC + + 9.0 version has 32 bit and 64 bit libraries.	

Using Libraries and Binaries Built with Visual Studio 2005 and Later

All MÄK products built with Microsoft Visual Studio require the C Runtime Library to function. The C runtime libraries have always been available from Microsoft for download, they are also installed on a user's machine when a Microsoft compiler is installed. The C runtime libraries are not part of the normal Windows installation. For customers who plan to use MÄK products on machines that do not have a compiler installed, MÄK has historically distributed a copy of the C Runtime Libraries with MÄK products. These libraries were put in the *bin* directory used by the MÄK products. MÄK products would then use the libraries in the *bin* directory and customers would not have a problem if copies of the libraries were not already installed.

Unfortunately, the C Runtime Libraries required by Microsoft Visual Studio 2005 (MSVC++8.0) and later cannot just be copied into the *bin* directory of an application. The libraries need to be installed correctly into Windows system folders. (The process is actually a little more complicated, a manifest file needs to be created to tell Windows where to find the libraries.)

To accommodate this change, MÄK distributes the Windows installer for the C runtime libraries with all MÄK products released for MSVC++8.0 and later. The 32-bit installer is named *vcredist_x86.exe*; the 64-bit installer (if supported) is named *vcredist_x64.exe*. They are in the base directory of any installed MÄK product that requires them.

Running the installer requires Administrator privileges for the machine the installer is run on.

For more information see this Microsoft URL:

<http://msdn2.microsoft.com/en-us/library/ms235299.aspx>



You must ensure that the preprocessor defines `_SECURE_SCL=0`, and `_HAS_ITERATOR_DEBUGGING=0` are set for MSVC++8.0 and MSVC++9.0 builds. If these are not set, random crashes and assertions may be encountered during runtime.

Compiler Compatibility on Windows

MÄK provides versions of product releases that have been compiled with Microsoft Visual C++ 7.1, 8.0, and 9.0. 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 the licensed version of MÄK RTI, you must install license management software. MÄK RTI 4.0.1 uses FLEXlm 11.8 for all versions except the Windows VC++ 7.1 version, which continues to use FLEXlm 11.6. If you are upgrading from a version of the MÄK RTI 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 MÄK RTI installer. You can download them at:

- ♦ Windows: <ftp://ftp.mak.com/out/flexlm11-8-win-setup.exe>
- ♦ Linux: <ftp://ftp.mak.com/out/flexlm11-8-linux.tar.gz>

A license manager FAQ is available at:

<http://www.mak.com/support/faq.php#license>

Third Party Library Dependencies

In addition to the Qt dependencies mentioned previously, the MÄK RTI 4.0.1 has the following dependencies:

- ♦ QT - 4.5 - <http://qt.nokia.com/>
- ♦ ZLIB - 1.2.3 - <http://www.zlib.net/>
- ♦ ICONV - 1.8 - <http://www.gnu.org/software/libiconv/>
- ♦ LIBXML2 - 2.6.22 - <http://www.xmlsoft.org/>

Patch Required for AMD Dual-processor Windows PCs

VR-Link-based products use a high resolution counter for time calculations on Windows PCs. Customers who are running Windows on PCs with multiple AMD Athlon 64-bit processors may notice clock jitter, which may cause time in MÄK products to run backwards. This occurs when the Windows scheduler changes the CPU the MÄK process is using. If the high resolution counters on each processor are not synchronized, the application may witness a decrease in the high resolution counter value stored in the processor causing an incorrect time calculation. To fix this problem customers, apply the AMD Dual-Core Optimizer patch provided by AMD. You can get the patch at:

http://www.amd.com/us-en/Processors/TechnicalResources/0,,30_182_871_9706,00.html



If you get an error when you try to access this URL, reload the page.

Backwards Compatibility

The MÄK RTI 4.0.1 libraries are not run-time compatible with previous versions. (In other words, you cannot have a federation execution in which some federates use MÄK RTI 4.0.1 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 4.0.1 is link compatible with previous versions, so a federate does not need to be recompiled to use this version. We recommend that all federates use the same version of the RTI when operating within the same federation (or using the same *rtiexec*).

Network Compatibility

The MÄK RTI is not network compatible with other RTIs. (In general, no two RTI implementations are network compatible, due to differences in the low-level network mechanisms that they use.) Applications that want to interoperate in the same federation execution must all use the same RTI implementation. A federation can easily switch from one RTI implementation to another between runs, but during each run, all participants must agree on which RTI to use, much as they must also agree on which FOM to use.

Compatibility with Other RTIs

The MÄK RTI 1.3 implements the API dictated by the 1.3 version of the HLA Interface Specification, as amended by DMSO's official HLA 1.3 Interpretations document. All RTIs that are verified as compliant with the HLA 1.3 specification are written to this exact API standard. This insures that the MÄK RTI is link-compatible with other RTIs that have been verified as HLA compliant.

The MÄK RTI 1516 implements the SISO DLC HLA API¹ 1516 (SISO-STD-004.1-2004). This API supports dynamic link compatibility, which was problematic with the original IEEE 1516 API. The IEEE 1516 API only supports compile time compatibility (uses C++ templates and implementation-specific header files). The MÄK RTI is compatible with any RTI written to the SISO DLC HLA API 1516.

Because an RTI is typically accessed by an application through a dynamic library (*.dll* on Windows, *.so* on UNIX), applications do not need to be recompiled or relinked to switch among dynamic-link-compatible RTI implementations that have been built using the same compilers. The only requirement is that the appropriate version of the library be located within the shared library search path. For more information, please see Section 3.2, “[Compiling and Linking with the MÄK RTI 1.3](#)” in *MÄK RTI Reference Manual*.

i

Because the DMSO RTI NG v6 was last built with the MSVC++ 6.0 compiler, it is not link compatible with the MÄK RTI 4.0.1. However, updated versions of the DMSO RTI provided by other vendors using compilers supported by MÄK RTI 4.0.1 would be link compatible.

1. Simulation Interoperability Standards Organization Dynamic Link Compatible High Level Architecture Application Program Interface

New Features and Updates

MÄK RTI 4.0.1 is a maintenance release. It has no new features. Two of the bug fixes are of critical importance for Linux users and are therefore discussed in this section.

- Linux federates built against MÄK RTI 4.0 are not link compatible with other versions of MÄK RTI. 42832

MÄK RTI 4.0 was released with versioned shared object files on Linux. As a result, any Linux federates that were built against MÄK RTI 4.0 are not dynamic link compatible with any other version of the RTI. All Linux users should upgrade to 4.0.1 and rebuild their federates against that version.

- The 64 bit HLA 1.3 libraries for Red Hat Enterprise 5 are not dynamic link compatible with other popular RTI implementations. 42638



We have updated our libraries to be compatible with these other RTIs. As a result, MÄK RTI 4.0.1, and all future RTI versions, will not be compatible with HLA 1.3 Linux federates built against MÄK RTI 3.4.1 or 4.0. These federates will need to be rebuilt against the new libraries.

Bug Fixes

The following bugs (with defect tracking numbers) have been fixed in this release:

- When using UDP forwarding, best-effort interactions were not being properly forwarded when fixed grid DDM was enabled. 42674
- A SyncPoint Messages function could potentially return a NULL pointer causing the federate ambassador callback announceSynchronizationPoint to crash. 42664
- The RTI was not correctly identifying scaffolding classes and was failing to merge valid FOM modules. 42633
- Shutting down an rtiexec and restarting disabled the options to shut down in the Federations View. 42601
- In some circumstances federates advancing time using nextMessageRequest would advance beyond the allowable time. 42659
- The Federation submenu in the Federations View would sometimes be blank when switching between connections. 42594
- Processing a federate on a lightweight connection did not show the correct HLA version in the Network Component View. 42604
- Resigning and restarting a federation would not correctly enable or disable the Option RTIsPy button in the Federations View. 42556
- The simpleDDM example did not compile out of the box on Linux. 42577
- The RTI did not run in any of the asynchronous IO processing models when using HLA Evolved. 42767

- ♦ The RTI failed to merge duplicate FOM modules when using the HLA_IMMEDIATE callback model. 42762
- ♦ The hash method of the Handle class in HLA Evolved was always returning 0. 42856
- ♦ The RTI was incorrectly treating the maximum value of an HLA Evolved HLAinteger64Time value as the maximum value of an unsigned integer 64. In order to preserve time implementation compatibility with HLA 1516, the previously misnamed *LogicalTimeImplInteger* class has been reimplemented as a signed 64 bit value rather than an unsigned. For 1516 federates that require an unsigned 64 bit integer implementation, a new class has been created called *LogicalTimeImplUnsignedInteger*, which functions exactly the same as the old *LogicalTimeImplInteger* class. 42874

Known Problems

This release has the following known problems:

- ♦ Distributed forwarder multicast subscription does not work for late-joining forwarders. With the introduction of the unified Distributed Forwarder a need arose to communicate between federates on a LAN and forwarders on remote LANs when a DM or DDM multicast group is joined or dropped, such that the forwarders can monitor or drop those groups on its LAN and forward messages back to the federates joined to such multicast groups. However the scheme to accomplish this does not work if the forwarder is not connected to the forwarder network when the federate joins the multicast group. The workaround is to ensure that the forwarder network is completely established prior to launching federates, or at the least, prior to allowing federates to attempt to join DM or DDM multicast groups.
- ♦ The MÄK Technologies Lisp (MTL) files used by MÄK products to specify configuration settings are parsed at startup by the RTI. When a configuration file has certain errors, such as unmatched parentheses, the parser will not necessarily return an error code. This can cause the RTI to reach a partially initialized state or cause the RTI to initialize itself with default RID values, causing unexpected behavior. This applies to lisp expressions in the RID file as well as the expressions passed to the 1516 createRTIambassador function.

- ♦ A workaround has been added to deal with situations where a host running the RTI Forwarder application has multiple network interface adapters (NICs) or has multiple IP addresses assigned to an interface, which may be the case if the RTI Forwarder host is connected to the WAN via a VPN tunnel. Under these circumstances the RTI Forwarder may fail to initiate connections to other forwarders if the IP address corresponding to its host in the Forwarder List section of the *routes.mtl* file does not match the IP address the RTI Forwarder application receives when it queries the host itself for its IP address. To correct this the IP addresses of the interfaces on which it will connect to other RTI Forwarders must be entered into the *routes.mtl* file Interface Address List section. At a minimum one such entry should correspond to the entry identifying the host in the Forwarder List section of the *routes.mtl* file. Please refer to Section 11.6, “[Configuring RTI Forwarders for Distributed Forwarding](#)” in *MÄK RTI Reference Manual* for details about the Interface Address List entry format.
- ♦ The HLA 1.3 version of the RTI can sometimes fail to throw an “ObjectAlreadyRegistered” exception if a federate tries to register an object with a name that is already being used by another object. The federate can register an object using a name that belongs to another object that is unknown to the local RTI component (LRC). The LRC only maintains the names of objects that are registered or discovered by the federate. While we plan to fix this problem, a good preventive measure is to always subscribe and tick the RTI before registering objects.
- ♦ The RTIspy browser-based GUI does not work well with Internet Explorer. While you will be able to see and use the RTIspy GUI with IE7, the page layout may be problematic, and some information may not be displayed correctly. The network map does not work in the default version of IE8. If you are using IE8, you may be able to view the network map in compatibility mode. For optimal use, MÄK recommends using Firefox (2.0 or later).

RTIspy Diagnostic GUI Scalability

The RTIspy diagnostic GUI provides helpful information and statistics that can be useful when debugging issues in federations with low federate counts, object counts, and update rates. It does not scale well to larger federations or federations with high frequency updates of many objects. Please contact MÄK if you require tools for debugging larger federations.

HLA Evolved API Issues

The HLA Evolved API has three functions for `createFederationExecution`. Unfortunately, two of them can be invoked in such a way that it is not clear which one is being used. The functions are as follows:

```
// variation 1
virtual void createFederationExecution (
    std::wstring const & federationExecutionName,
    std::wstring const & fomModule,
    std::wstring const & logicalTimeImplementationName = L"");

// variation 2
virtual void createFederationExecution (
    std::wstring const & federationExecutionName,
    std::vector<std::wstring> const & fomModules,
    std::wstring const & logicalTimeImplementationName = L"");

// variation 3
virtual void createFederationExecution (
    std::wstring const & federationExecutionName,
    std::vector<std::wstring> const & fomModules,
    std::wstring const & mimModule,
    std::wstring const & logicalTimeImplementationName = L"");
```

Variation one is unambiguous. However, consider the following code:

```
createFederationExecution( L"FedExName", vectorOfFomModules, L"");
```

It is not clear whether you are using variation two or three, and the compiler will throw an error. To avoid this error we recommend that you always use variation three. If you do not want to specify a `mimModule`, then pass in an empty string.

