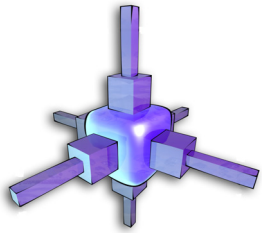

MÄK RTI 3.3.2 Release Notes

This document provides the following release-specific information for MÄK RTI 3.3.2:

MÄK RTI 3.0 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).)

Supported Systems

Table 1 lists the platforms supported by the standard version of MÄK RTI 3.3.2. 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	Default compiler.
Red Hat Fedora 7	Default compiler.
Windows XP	Microsoft Visual C++ 7.1, 8.0, 9.0**
Windows Vista*	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.	

The RTI LS supports federates that run on the Solaris and IRIX operating systems. (Please see *MÄK RTI Release Notes* for specific versions supported.) This version of the MÄK RTI only includes the LRC libraries. It does not include the rtiexec, RTIspy, or distributed forwarder. You can link to the libraries if you are creating new federates on IRIX or Solaris.

If you need to run an rtiexec, you must do so on Windows or Linux.

Table 2: RTI LS: Platforms supported

Operating System	Compiler
SGI IRIX6.5	MipsPro7.4.2m C++ compiler
Sun Sparc Solaris 10	Sun C++ 5.8

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, with the release of the new C Runtime Libraries required by Microsoft Visual Studio 2005 (MSVC++8.0) and later, the libraries can no longer 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 is distributing 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. MÄK has chosen to not integrate the MÄK installer and the Microsoft installer so as not to require users to have Administrator privileges to install MÄK products. Therefore, if you who do not have a compiler installed, or get error messages like "Software has not been installed correctly, please re-install", you must apply the patch.

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 can result in program instability.

FLEXlm Support

MÄK RTI 3.3.2 uses FLEXlm 11.6 for Windows, Linux, and Sun Solaris. SGI IRIX uses FLEXlm 9.2.

Third Party Library Dependencies

In addition to the Qt dependencies mentioned previously, the MÄK RTI 3.3.2 has the following dependencies:

- QT - 4.4.1 - http://www.trolltech.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/
- VR-Link 3.12
- EHS - 1.3.1 (modified) http://xaxxon.slackworks.com/ehs/
- PCRE - 6.4 (modified)) http://www.pcre.org/
- PME - 1.0.4 (modified) http://xaxxon.slackworks.com/pme/index.html
- PTHREADS - 2.7.0 http://sourceware.org/pthreads-win32/.

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.

RTIspy Web-based GUI Configuration

The RTIspy has the following system considerations:

- The RTIspy is optimized for screen resolutions of 1024x768 and larger. For smaller screen sizes, additional window scrolling will be required.
- The RTIspy has been tested and will display correctly with Mozilla Firefox version 1.5 and higher and Internet Explorer versions 6 and higher. Other browsers may be compatible, but they have not been tested and are not specifically supported.

Backwards Compatibility

The MÄK RTI 3.3.2 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 3.3.2 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 3.3.2 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 API[1] 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 4.2, "Compiling and Linking with the RTI 1.3" in *MÄK RTI Reference Manual*.

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

Because the DMSO RTI NG v6 was last built with the MSVC++ 6.0 compiler, it is not link compatible with the MÄK RTI 3.3.2. However, updated versions of the DMSO RTI provided by other vendors using compilers supported by MÄK RTI 3.3.2 would be link compatible.

New Features and Updates

MÄK RTI 3.3.2 is primarily a maintenance release. It also has the following minor updates:

- Support for save and restore between 1.3 and 1516 federates. Also, you no longer need to enable RTI_forceFullCompliance to use save and restore.
- The following RID file parameters have been added:
  - **RTI_enableSaveRestoreWhenUsingRtiexec** - Enables (1) or disables (0) the save and restore services when using an rtiexec. Default = 1
  - **RTI_dumpRid** - Enables (1) or disables (0) dumping of the contents of the RID file when the notification level is set to notify (>2). Default: 0.
- Upgrade to FLEXlm 11.6. You must replace the files on your license server with the FLEXlm 11.6 files. Your current license is forward compatible.

Bug Fixes

The following bugs have been fixed in this release:

- On Linux, the RTI Assistant occasionally continuously displayed an error indicating “Too many open files”.
- If the asynchronous callbacks processing model (RTI_processingModel = 3) was used, there were no federate ambassador callbacks for both object updates and interactions.
- The RTI crashed when registering an object with a region without supplying a name when using fixed grid DDM.
- The RTI crashed on exit if a DDM region was created, used for a subscription, and then deleted without ever committing any changes to it using the notifyAboutRegionModification service.
- The **RTI_networkInterfaceAddr** RID parameter did not work.
- RTI Assistant notifications were displayed in the wrong location when using VC9.
- Fixed some minor bugs related to updating the RTI Assistant connection query dialog boxes when new connections were found.

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 RID consistency feature relies on the ability to establish a connection between a federate and the RID tester (presumably the rtiexec), however if the federate's **RTI_enableTcpCompression** parameter and the RID tester's parameter are different, the two cannot communicate, and consequently the RID consistency test request and response messages cannot be exchanged. The workaround is to ensure that the **RTI_enableTcpCompression** parameter is consistent among all participants prior to launch.
- 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 13.7, "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.
- To use shared memory on operating systems other than Windows, you must start the Shared Memory Queue Manager manually. Automatic launching of the shared memory queue manager has been disabled because the current implementation resulted in a conflict for console I/O.
- The RTI Dump utility does not work directly with shared memory only mode. It can be used with shared memory if the shared memory mode relays messages to the network and the RTI Dump utility is connected via the network communications.
- Smart Forwarding does not work when using fixed grid DDM. The smart forwarding feature relies on the exchange of DM and DDM information in order to establish a correct destination list. Because fixed grid DDM does not exchange DM or DDM information, enabling smart forwarding causes no messages to be sent because there are no destinations.
- When merging FOM modules, the dimensions are taken from the create FOM module and no others are merged.
- The RTIspy browser-based GUI does not work well with Internet Explorer 6 or 7. 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. For optimal use, MÄK recommends using Firefox (2.0 or later).
- Using the MÄK RTI on Fedora Core 7 machines that have Xen Virtualization software installed causes a problem with multicast discovery. This is because INADDR_ANY maps to the virtual interface rather than the default, eth0. To resolve this issue, set **RTI_mcastDevAddrString** to the IP address of the real ethernet card.
- If you use shared memory, you cannot use the RTI Assistant to configure a connection. To use shared memory, you must set the **RTI_configureConnectionWithRid** to 1.
- The Search feature in online help does not work. This is due to a bug in the Qt Assistant browser.

RTI 1516 Implementation Issues

The MÄK RTI 1516 has the following known issues:

- While the 1516 API uses the std:wstring to exchange string information, internally, the MÄK RTI 1516 operates on narrow strings only (that is, representable by std::string). Therefore, the contents of any wide string passed through the RTI API must be convertible to a narrow string.
- The publish and subscribe services in the 1516 interface specification are additive when successive calls are made, versus replacement for the 1.3 interface specification. The MÄK RTI 1516 implementation follows the additive policy; however, the MÄK RTI 1.3 follows the replacement policy. As a result, the advisories and discoveries may conflict when a federation is composed of both 1.3 and 1516 federates. The **RTI_processUnknownUpdatesForDiscovery** RID parameter should be enabled as a work around for discoveries.