MÄK RTI 2.4 Release Notes

This document provides the following release-specific information for MÄK RTI 2.4:

Document ID: RTI-2.4-2-050715

Supported Systems

Table 1 lists the platforms supported by the MÄK RTI 2.4. 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: Platforms supported

Platform	Compiler
SGI IRIX6.5	MipsPro7.3 C++ compiler (available from SGI)
Sun Sparcstation with Solaris2.7 or later	SPARCompiler C++ version 5.2 (available from Sun)
PC with Red Hat Linux 9.0	gcc version 3.2.2
PC with Red Hat Enterprise 3	gcc version 3.2.3
PC with Red Hat Fedora Core 3	gcc version 3.4.3
PC with Red Hat Enterprise 4	gcc version 3.4.3
PC with Windows 2000/XP	Microsoft Visual C++ 6.0 and 7.1 (.NET)

MÄK RTI 2.4 was built with VR-Link 3.9.5.

Qt Release Compatibility

The MÄK RTIspy GUI uses Trolltech's Qt cross-platform GUI toolkit. In particular, the GUI included with MÄK RTI 2.4 relies on Qt version 3.3.1. (The Qt run-time libraries are distributed with the MÄK RTI). If a federate also uses Qt, it must be sure to use the same version of Qt as the MÄK RTIspy GUI, in order for the RTIspy GUI to function properly for that federate. Federates built with other versions of Qt should disable the RTIspy GUI to avoid these incompatibility problems.

Patching the Microsoft Visual C++ 6.0 Compiler

If you use Microsoft Visual C++ version 6.0, you must install a patch to a standard header file, in order to successfully build applications for HLA 1.3 or 1516. The patch comes from Dinkumware, Ltd (the company that developed the Standard C++ Library header files for Microsoft), but for convenience, we are distributing it with MÄK RTI releases.

The patch fixes a bug that would cause applications to crash if they pass std::sets or std::maps from an application to a DLL or vice-versa. For details about the patch go to http://www.dinkumware.com/vc_fixes.html.

To install the patch:

1. In the top level MÄK RTI directory, is a file called *XTREE-msvc++patch*. Rename this file *XTREE*.
2. Make a back-up version of the original *XTREE*.
3. In the Visual Studio *include* directory (for example, *C:\Program Files\Microsoft Visual Studio\VC98\Include*), replace the original version of *XTREE* with the patched version.

Compiler Compatibility on Windows

MÄK provides versions of product releases that have been compiled with Microsoft Visual C++ 6.0 and 7.1. When you run MÄK products together, for example, the MÄK RTI and the Stealth, 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.

Backwards Compatibility

The MÄK RTI 2.4 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 2.4 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 2.4 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 (which include, but are not limited to packet layout.)) 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 the DMSO RTI

With the exception of several additional functions (such as, DtReadFileDescriptor(), described in MÄK RTI documentation), the MÄK RTI 2.4 for HLA 1.3 has the exact same API as the DMSO RTI 1.3 NG. Unless you are using the RtReadFileDescriptor function, an application built against the DMSO RTI NG does not need to be recompiled in order to be used with the MÄK RTI 2.4, or vice versa (except that the Linux version is not compatible with the DMSO RTI).

MÄK RTI 2.4 uses the same FED file format as the DMSO RTI, but a different RID file format. (RID file formats depend on the RTI-implementation.)

New Features and Updates

MAK RTI 2.4 has the following changes and new features:

- MÄK RTI 2.4 adds Java bindings for HLA 1516.
- Improved interoperability betwen the 1.3 and 1516 versions of the RTI
- Distributed TCP forwarding
- Multicast discovery of TCP Forwarders and the rtiexec
- Changes to configuration for DM class multicast filtering
- Asynchronous callbacks.
- Changes to configuring RTIspy
- New configuration parameters
- Changes to log file naming conventions
- Miscellaneous API changes
- An RTI implementation issue regarding MOM has been removed.
- Online help for RTIspy.

Java Bindings for HLA 1516

This release adds Java bindings for the HLA 1516 version of the MÄK RTI. For isntallation instructions, see chapter 2 in *MÄK RTI Reference Manual*.

Improved 1.3 and 1516 Interoperability

It is now possible for a 1.3 federate and a 1516 federate to interoperate using their native FOM file format (for example, FED and FDD respectively). The files must specify the same classes, attributes, and parameters excluding the MOM classes.

Interoperability of the MOM is not supported between 1.3 and 1516.

It is now possible for a 1.3 federate and a 1516 federate to interoperate using the DDM services. The dimensions in the 1.3 routing spaces are placed into a global routing space. As a result, all dimensions with the same name across different routing spaces resolve to a single dimension. This feature is supported by enabling the RTI_extend13and1516interop RID parameter.

Distributed TCP Forwarding

The MÄK RTI now supports the use of multiple TCP Forwarders via its Distributed TCP Forwarding feature. Distributed TCP forwarding can greatly reduce the bandwidth consumed by federations operating across a WAN by sending the absolute minimum number of messages between LANs. It reduces the number of messages sent across a WAN to either 1 copy per LAN or 0 if Smart Forwarding is enabled and no federates on the remote LAN require the message. Distributed TCP Forwarding can also be used to greatly improve scalability for very large federations when large volumes of reliable messages are sent within the federation by creating a hierarchy of forwarders.

Distributed TCP forwarding is supported via the *distributedForwarder.exe* application in *./bin*. The *distributedForwarder.exe* application is also an alternative to *tcpForwarder.exe* application for standard TCP forwarding and uses the same RID configuration settings.

See *MÄK RTI Reference Manual* for more details on Distributed TCP Forwarding and running *distributedForwarder.exe*.

Multicast Discovery for TCP Forwarders and the rtiexec

The MÄK RTI now supports automatic discovery of the local TCP Forwarder and the rtiexec. With multicast discovery enabled, federates will automatically detect the TCP Forwarder and rtiexec running on their local network. So with multicast discovery enabled, it is no longer necessary to specify the **RTI_tcpForwarderAddr** for TCP Forwarders running on the same LAN as the federate. When reliable internal messaging is disabled, multicast discovery also ensures that only a single RTI Exec can run on a LAN with the same [**RTI_udpPort**, **RTI_destAddrString**] pair. This makes it easier to start and stop fecerations using scripts and eases RID configuration greatly. See *MÄK RTI Reference Manual* for more details on Multicast Discovery.

Configuring DM Class Multicast Filtering

The format for specifying **RTI-addDMObjectMulticastAddr** and **RTI-addDMInteractionMulticastAddr** has changed. In addition to FOM class, multicast address, and nesting, you must now specify a destination device for the multicast traffic. If you have multiple network cards, this allows you to use of both networks by dividing RTI network traffic between them. By default, the network interface is set to "0.0.0.0" or nil which instructs the RTI to use the default network interface. (This feature was available in prior releases, but the parameter was previously optional.)

```
;; DM Class Multicast Filtering
;; These settings use separate multicast addresses
;; to transmit/receive object and interaction data
(RTI-addDMObjectMulticastAddr "ObjectRoot" "224.0.0.1" "inclusive"
  "0.0.0.0")
(RTI-addDMInteractionMulticastAddr "InteractionRoot" "224.0.0.2"
  "inclusive" "0.0.0.0")
```

Support for Asynchronous Callbacks

The MÄK RTI now supports an asynchronous callback mode. Instead of buffering messages to be processed during the next RTI tick, the asynchronous thread processes the messages directly. In this mode, the federate need not invoke tick, because all internal RTI processing is performed in the RTI Ambassador calls or the asynchronous thread. This configuration requires the Federate Ambassador callbacks to be thread safe. This feature is enabled through the **RTI_asynchronousCallbacks** RID parameter.

Changes to Configuring RTIspy

The RTIspy LRC GUI plug-in is now installed automatically with the standard MÄK RTI installation. To enable the RTIspy LRC GUI, set the **RTI_enableLrcGUI** parameter to 1.

RID files that use the **RTI-addPluginName** method of loading the LRC GUI are still supported.

New Configuration Parameters

The MÄK RTI has the following new configuration parameters:

- **RTI_forwardingDelay** – When publication changes affect forwarding, routing filters are ignored for the specified interval to allow updates to remote subscription information.
- **RTI_mcastDiscoveryInterval** – The interval to wait for a discovery response.
- **RTI_mcastDiscoveryEnabled** – If mcastDiscoveryEnabled is enabled, then the federate will attempt to locate the local TCP Forwarder automatically and it is not necessary to set RTI_tcpForwarderAddr manually.
- **RTI_mcastDiscoveryAddrString** – The multicast channel used for TCP Forwarder discovery.
- **RTI_mcastDiscoveryTries** – The number of attempts at multicast discovery to attempt before failing discovery when no response is received.
- **RTI_mcastDiscoveryInterval** – The interval to wait for a discovery response.
- **RTI_enableLrcGUI** – Set to 1 to enable the RTIspy LRC console GUI.
- **RTI_forwarderRoutesFile** – Set the name for the routing configuration file for the Distributed TCP Forwarder.
- **RTI_asynchronousCallback** – Process messages in an asynchronous thread. The federate must be able to receive Federate Ambassador callbacks asynchronously. (This parameter forces asynchronous I/O.)
- **RTI_extend13and1516interop** – Extends interoperability between 1.3 and 1516 (for example, handling DDM dimensions.)
- **RTI_enableFomBackwardsCompatability** - If compatability between HLA 1.3 FED files and HLA 1516 FDD files is not required in the federation, you can enable **RTI_enableFomBackwardsCompatability** to instruct the RTI to assign FOM handles according to the pre-RTI 2.4 numbering scheme. This is useful only for federates that rely on consistant FOM handles from run to run (this behaviour is not required by the HLA specification, and therefore should not be relied upon). It disables **RTI_extend13and1516interop**.

During a federation save, it is possible that the federation save will be initiated at a federate while messages are in transit or still being generated by another federate. In this case, the MÄK RTI has buffered the messages and saved them to be delivered after the federation saved has been restored. This behavior can now be controlled using the following two parameters:

- **RTI_deliverTransientMessagesDuringSave** – Messages received by the LRC during a save are delivered to the federate (for example, discover, remove, reflect, receiveInteraction.)
- **RTI_saveTransientMessages** – If transient messages are not delivered during a save, save transient messages to be delivered upon restore.

Log File Naming Conventions

When **RTI_logFileName** is enabled and **RTI_reuseLogFile** is disabled, the RTI will now insert the unique ID for the log file before any filename extension. For example if **RTI_logFileName** = "makRti.log", federates will generate logs with unique names such as: "makRti1120679916.log". In previous versions, the same log file would have been named "makRti.log1120679916". Injecting the ID before the extension allows the log files to work with Windows file type detection.

Miscellaneous API Changes

The exception InvalidFederateHandle was added to the 1516 API. This exception is used in the normalizeFederateHandle service call.

An integer time implementation, LogicalTimeImplInteger, was added to the 1516 logical time library. An integer implementation is now preferred as the floating point implementation is known to cause anomalous message ordering due to imprecision in floating point comparisons. The floating point time implementation was renamed to LogicalTimeImplFloat. The LogicalTimeImpl (as a float implementation) is maintained in the library for now but its use is deprecated.

MOM Implementation

An implementations issue in the 1516 version of the RTI regarding the format of strings in the MOM object attributes and interaction parameters has been resolved.

Online Help

The RTIspy GUI windows now have online help.

Documentation Updates

MÄK RTI Reference Manual has been updated for this release.

Bug Fixes

The following bugs have been fixed in this release:

- This release fixes a memory leak present in previous 2.3.x releases.
- For 1516, the format of strings in MOM object attributes and interaction parameters now uses the correct UTF-16 formatting.
- The DDM default ranges are no longer ignored when constructing region realizations.
- The TCP forwarder now correctly sets the socket buffer sizes as specified by the **RTI_socketReceiveBufferSize** and **RTI_socketSendBufferSize** RID parameters.
- On certain systems the rtiexec could become unstable and stop allowing federates to connect after running for some time. This issue has been resolved.
- The rtiexec GUI now runs in a separate thread from the main rtiexec processing to essentially eliminate its impact on rtiexec performance. Previously, use of the GUI for the rtiexec affected the performance of algorithms requiring centralized control such as time management and save/restore.
- In the 1516 version, synchronization point tags are saved and restored. This issue was fixed in a previous release, but had not been noted as being fixed.

Known Problems

This release has the following known problems:

- 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.
- If you are using the Linux version of the MÄK RTIspy with versions of MÄK products earlier than the most recent releases, the MÄK products may crash. If this happens, contact MÄK technical support and we will provide you with a fix for your application.
- The rtiexec does not connect the output streams to a console until after reading the RID file in Windows. As a result, errors opening or parsing the RID file are not displayed. If **RTI_enablePopUpErrorMsgs** is enabled then these errors are displayed in a popup box, however this solution works only under Windows.

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.