MÄK RTI 2.3 Release Notes

This document provides the following release-specific information for MÄK RTI 2.3:

Document ID: RTI-2.3-2-041006

Supported Systems

Table 1 lists the platforms supported by the MÄK RTI 2.3. 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	GNU C++ (g++) distributed with Red Hat distribution. gcc version 3.2
PC with Windows 2000/XP	Microsoft Visual C++ Microsoft .NET

MÄK RTI 2.3 was built with VR-Link 3.9.3.

Qt Release Compatibility

The MÄK RTIspy GUI uses Trolltech's Qt cross-platform GUI toolkit. In particular, the GUI included with MÄK RTI 2.3 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 against the IEEE 1516 header files. 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, something that IEEE 1516 federates must do. 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.

Backwards Compatibility

The MÄK RTI 2.3 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 2.3 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 2.3 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 2.3 for HLA 1.3 has the exact same API as the DMSO RTI 1.3 NG (except v5). 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 2.3, or vice versa (except that the Linux version is not compatible with the DMSO RTI).

MÄK RTI 2.3 uses the same FED file format as the DMSO RTI, but a different RID file format. (RID file formats depend on the RTI-implementation.)

MÄK products, including the MÄK RTI 2.3, are not compatible with the DMSO RTI NG v5 due to changes in the how the DMSO RTI handles namespaces.

New Features and Updates

MÄK RTI 2.3 has the following new features and updates:

- The RTI processes the switches table when reading the FOM in the FDD file format to set the default settings of the federation switches.
- RTI diagnostics can be logged to a file.
- 1516 implementation issues
- RTIspy updates
- New configuration parameters
- The RTIspy GUI components use Qt 3.3.1
- TCP forwarder utility
- Smart forwarding for internal messages
- RTIdump utility supports reliable traffic.

RTI Diagnostics can be Logged to A File

In the LRC, the **RTI_logfileName** parameter enables logging to a file with the specified name. In the rtiexec, the command line option `-l filename` enables logging to a file with the specified name.

RTI 1516 Implementation Issues

This initial implementation of the 1516 services resulted in some deficiencies with respect to the HLA 1516 specification. It is expected that these deficiencies will be rectified as the MAK RTI goes through the RTI verification process.

The following issues were resolved:

- The 1516 LogicalTime, LogicalTimeInterval, and LogicalTimeFactory destructors where moved from the logical time library into the RTI library.
- The 1516 MOM support was improved to include use of MOM DDM regions and response to additional MOM attributes and interactions added by 1516.
- The federate ambassador synchronizationPointRegistrationFailed service call returns the appropriate value.
- The federate ambassador federationNotSaved service call returns the appropriate value.
- The federate ambassador federationNotRestored service call returns the appropriate value.
- The queryFederationSaveStatus and queryFederationRestoreStatus are fully implemented. The corresponding federationSaveStatusResponse and federationRestoreStatusResponse will be called appropriately.
- The tags supplied in the requestAttributeValueUpdate service calls are carried through to the provideAttributeValueUpdate service calls.

- The MOM auto provide switch is implemented.

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 format of strings in the MOM object attributes and interaction parameters use a simple memory copy of the wchar_t arrays returned by std::wstring::c_str(). As a result, these attributes and parameters are not compatible across all platforms. For example, the Microsoft Visual Studio compiler stores the wchar_t type as two bytes while the Gnu C++ compiler stores the wchar_t type as four bytes.
- The synchronization point tags are not saved nor restored.
- 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. This issue will be resolved in subsequent releases to apply the appropriate policy to publish and subscribe operations.

RTIspy API Changes

The following changes have been made to the RTIspy API.

- Message throttle methods were added to the *DtConnectionMgr* class. These methods allow the LRC to activate message delays and throttle message transmissions. Associated methods were added to the interaction, object, and DDM managers to control message throttling.
- Mutators and accessors to manipulate the save/restore status were added to the *DtFederate* class of the federation execution.
- The methods to set and get the FOM type from the *DtFomClassMgr* class were moved up into the parent VR-Link class *DtFedTarget*. Methods to set and get switch table settings were added to the *DtFedTarget* class.
- The methods for manipulating the bit mask routing in the *DtRtiMsg* class were modified.
- An anonymous parameter was added to the *DtInteractionMgr* (and subclasses) to support sending MOM interactions using regions. Similarly, an update type parameter was added to the *DtObjMgr* (and subclasses) to support sending MOM updates using regions.
- The setFromPhvps() member function in the MOM *DtMomInteraction* class and all derived subclasses now returns a boolean that indicates whether decoding the parameters creates a valid interaction.

- The MOM *DtMomSR* class supports updating static, conditional, and periodic attributes. A map from element handles to attribute handles was also added. The opposite mapping from attribute handles to element handles already existed.
- The MOM *DtMomPublisher* class supports sending updates using a DDM region.

TCP Forwarder Utility

Two new executables *tcpForwarder.exe* and *tcpForwarder1516.exe* have been added to the *makRti/bin* directory.

The TCP Forwarder utility allows you to improve RTI performance by separating the TCP Forwarder process from the rtiexec process. By disabling the **RTI_rtixecHasTcpForwarderThread** parameter, you can instruct the rtiexec not to create an internal TCP Forwarder. If the TCP Forwarder utility is then run on a separate machine from the federates and the rtiexec, overall federation performance may improve significantly.

Smart Forwarding for Internal Messages

The TCP Forwarder can now be instructed to send internal messages only to destinations that could potentially be interested in them. To make use of this feature, enable the new **RTI_enableFedexMsgRouting** parameter.

This feature can significantly reduce the number of internal message transmissions, reducing both bandwidth and network overhead. In those federations in which internal RTI messages consume a significant portion of network resources this can lead to a substantial, federation-wide performance improvement.

rtiDump Utility for Reliable Traffic

The Rtidump utility has been updated to sniff and report reliable traffic as well as best-effort traffic. You can use it to listen to both reliable FOM data and internal RTI messages when the **RTI_internalMsgReliable** parameter is enabled.

New and Updated Configuration Parameters

The MÄK RTI has the following new configuration parameters:

- **RTI_transmitDelay** – specifies the delay time for processing internal messages
- **RTI_transmitRate** – specifies that internal messages should be sent in one tick or across ticks
- **RTI_logfileName** – specifies a log file for diagnostic messages
- **RTI_rtiExecHasTcpForwarderThread** – allows you to run the TCP forwarder in an internal thread
- **RTI_forceFomDataReliable** – specifies transmission of FOM data via the reliable transport channel
- **RTI_enableFedexMsgRouting** – applies basic forwarding rules to internal messages
- **RTI_enablePopUpErrorMsgs** – displays pop-up messages for some RTI errors (Windows only)
- **RTI_bestEffortSendRetries** – provides congestion control
- **RTI_bestEffortSendRetryWaitUse** – provides congestion control
- **RTI_socketSendBufferSize** – specifies the socket send buffer size
- **RTI_enableRtiexecGUIConsoleLog** – enables the rtiexec GUI log.

The following parameters have been updated:

- **RTI_tcpBufferSize** – increased the default buffer size
- **RTI-addDMObjectMulticastAddr** – added optional parameter to assign DM Multicast groups to specific network interfaces
- **RTI-addDMInteractionMulticastAddr** – added optional parameter to assign DM Multicast groups to specific network interfaces.

Configuring the Delay Time for Processing Internal Messages

The **RTI_transmitDelay** parameter specifies that requests are processed immediately (the default) or after a specified delay (in seconds).

During federation startup it is typical for federates to perform declaration and object management services that requires the RTI to exchange a significant number of internal messages. Often, these messages are redundant, simply satisfying duplicate requests from each joining federate. For example, the LRC of a federate that has registered an object will resend the register information for each subscription message that it receives (that matches the registered object). The LRC can condense these duplicate responses (for example, register messages) by delaying the processing of requests (for example, subscription messages) allowing a single response to satisfy multiple requests. Another example is the processing of request attribute values. By delaying the invocation of the federate ambassador provide attribute value service, a single invocation can handle the multiple requests typically occurring during federation startup. The caveat for the 1516 API is that the user supplied tag is dropped.

Transmitting Internal Messages Across Ticks

The RTI_transmitRate parameter specifies that the LRC should transmit all required internal messages during a single tick (the default) or that it should transmit internal message responses across multiple ticks at the specified rate (based on time since last tick.)

During federation startup it is typical for federates to perform declaration and object management services that require the RTI to exchange a significant number of internal messages. Often, the LRC has to respond to messages that it receives by retransmitting messages about its internal state (for example, object registrations and class publications). The LRC retransmit all the necessary messages during a single tick. This message retransmission process can cause a spike in the message bandwidth. To smooth out these message spikes, the LRC can be throttled to retransmit message at a specified rate across subsequent ticks.

The throttle only affects the retransmission of internal messages during tick. It does not affect message required by services invoked by the federate (for example, updates and interactions).

Logging Diagnostic Data to A File

The RTI_logfileName parameter enables or disables (default) logging of diagnostics to a file.

The RTI can log its diagnostic output to a file. To disable logging, specify an empty string (""). To enable logging, specify a name for the log file.

Running the TCP Forwarder in an Internal Thread

The RTI_rtiExecHasTcpForwarderThread parameter enables (default) or disables running a TCP Forwarder in an internal thread. When using the new tcpForwarder utility, this parameter may be disabled to improve RTI Exec efficiency.

Transmitting FOM Data via Reliable Transport

The RTI_forceFomDataReliable parameter enables or disables (default) transmission of all FOM data via the reliable transport channel. You must run the rtiexec to enable this feature.

Applying Basic Forwarding Rules to Internal Messages

The RTI_enableFedexMsgRouting parameter enables or disables the TCP Forwarder to apply basic forwarding rules to internal messages. This will significantly reduce internal network traffic.

Displaying Pop-Up Error Message Windows (Windows only)

The RTI_enablePopUpErrorMsgs parameter enables or disables display of pop-up error message windows for certain classes of RTI error, which are otherwise reported only as RTIinternalError exceptions. This feature is only available under the Windows operating system.

Controlling Congestion for Best-Effort Sockets

RTI_bestEffortSendRetries and RTI_bestEffortSendRetryWaitUse are advanced network configuration parameters. When used together, they provide limited congestion control for Best-Effort sockets. RTI_bestEffortSendRetries determines the number of send attempts before discarding the message. RTI_bestEffortSendRetryWaitUse determines the number of microseconds to wait after each send attempt for a blocked socket.

Specifying the Socket Send Buffer Size

The RTI_socketSendBufferSize parameters specifies the socket send buffer size.

Enabling the rtiexec GUI Console Log

The RTI_enableRtiexecGUIConsoleLog parameter enables (1) or disables (0) piping of MÄK notification streams (DtWarn, DtInfo, and so on) to the rtiexec GUI console log. Default: 0.

New Default TCP Buffer Size

The default TCP buffer size has been increased from 20 KB to 50 KB to improve RTI performance for large messages and bundles. For many platforms, it will also be useful to increase RTI_socketReceiveBufferSize to 100 KB. Unfortunately such large buffer sizes are not supported on all platforms.

Assigning DM Multicast Groups to Network Interfaces

The RTI-addDMObjectMulticastAddr parameter has a new optional parameter that assigns DM Multicast groups to specific network interfaces. If you have multi-homed machines, this parameter allows you to segment RTI traffic onto multiple physical networks without modifying your system's routing tables. For example, if a multi-homed machine has network interfaces with address 10.0.1.211 and 192.168.1.211, you can send all HLA object traffic to the 10.0.1.x network as follows:

```
(RTI-addDMObjectMulticastAddr "ObjectRoot" "224.0.0.1" "inclusive"
  10.0.1.211)
```

Assigning DM Multicast Groups to Network Interfaces

The **RTI-addDMInteractionMulticastAddr** parameter has a new optional parameter that assigns DM Multicast groups to specific network interfaces. If you have multi-homed machines, this allows you to segment RTI traffic onto multiple physical networks without modifying your system's routing tables. For example, if a multi-homed machine has network interfaces with address 10.0.1.211 and 192.168.2.211 you can send all HLA interaction traffic to the 192.168.1.x network using the following setting:

```
(RTI-addDMInteractionMulticastAddr "InteractionRoot" "224.0.0.2"
  "inclusive" 192.168.2.211)
```

Documentation Updates

Note the following updates to *MÄK RTI Reference Manual*:

- The printed version of *MÄK RTI Reference Manual* states that the MÄK RTI comes with, or is included with, all MÄK products. These statements mean that the MÄK RTI software is on product CDs. It does not mean that purchase of a MÄK product includes a MÄK RTI license. The MÄK RTI is licensed separately from other MÄK products.
- The *MÄK RTI Reference Manual* has not yet been updated to reflect support for HLA 1516.
- The following note has been added to "Starting the rtiexec," on page 3-7, in *MÄK RTI Reference Manual*:

! The *rtiexec* is a server application. Do not run more than one *rtiexec* per federation.

- The RTIspy Network Monitoring Tool is based in part on the work of the Qwt project (http://qwt.sf.net).

Bug Fixes

The following bugs have been fixed in this release:

- RTI API class reference files are included in this release.
- The RTI Exec Plugin API was difficult to use under Windows in previous releases. To fix this, the RTI Exec now links to the RTI library dynamically instead of statically.
- A problem with the rtiexec console's time state view, which sometimes resulted in RTI crashes for time managed federations, has been fixed.
- The QT DLL in release 2.2 was built with debug libraries. In this release, it no longer requires debug libraries.

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.