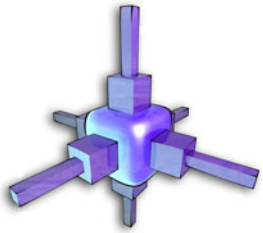

MÄK RTI 3.0 Release Notes

This document provides the following release-specific information for MÄK RTI 3.0:

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).)

Document ID: RTI-3.0-2-060330

Supported Systems

Table 1 lists the platforms supported by the MÄK RTI 3.0. 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	MipsPro7.4.2m C++ compiler
Sun Sparcstation with Solaris 8	Sun Workshiop 6 with C++ 5.
Red Hat Enterprise Linux Workstation 3	GNU C++ (GCC) 3.2.3; glibc 2.3.2
Red Hat Enterprise Linux Workstation 4	GNU C++ (GCC) 3.4.4; glibc 2.3.4
Red Hat Fedora Core 3	GNU C++ (GCC) 3.4.4; glibc 2.3.6
PC with Windows 2000/XP	Microsoft Visual C++ 6.0 and 7.1

MÄK RTI 3.0 was built with VR-Link 3.9.6.

Qt Release Compatibility

The MÄK RTIspy GUI uses Trolltech's Qt cross-platform GUI toolkit. (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.

The default RTIspy LRC Console GUI Plug-in is built with Qt 3.3.5. For federates such as recent versions of the MÄK Logger, MÄK Stealth, and VR-Forces 3.8, which require Qt 3.3.1, an alternate build of the RTIspy GUI Plug-in is included with MÄK RTI 3.0. The Qt 3.3.1 versions of the plug-in for Windows are:

- *LRCconsole-qt3.3.1.dll*
- *LRCconsoled-qt3.3.1.dll*
- *LRCconsole1516-qt3.3.1.dll*
- *LRCconsole1516d-qt3.3.1.dll*.

The Qt 3.3.1 versions of the plug-in for UNIX are:

- *LRCconsole-qt3.3.1.so*
- *LRCconsole1516-qt3.3.1.so*.

To select the Qt 3.3.1 version of the RTIspy LRC Console GUI instead of the Qt 3.3.5 version, disable the **RTI_enableLrcGui** RID file parameter and use the **RTI-addPluginName** parameter. For example, to select the debug build of the Qt 3.3.1 plug-in for HLA 1.3 set the following RID parameters:

```
(setqb RTI_enableLrcGUI 0)
(RTI-addPluginName "LRCconsole-qt3.3.1.dll")
```

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 on the same computer, 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 3.0 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 3.0 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 3.0 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 3.0 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 3.0, or vice versa (except that the Linux version is not compatible with the DMSO RTI).

MÄK RTI 3.0 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 3.0 has the following new features and updates:

- Verification of the 1516 version
- Optimizations
- New configuration parameters
- New example and example documentation
- Fallback to lightweight mode is no longer supported
- Improved installation process.

HLA 1516 Verification

The MAK RTI has been formally verified by the Defense Modeling and Simulation Office (DMSO) as fully compliant with the IEEE 1516 version of the HLA Interface Specification. Our verification status can be viewed on DMSO's RTI Verification Status Board:

https://www.dmso.mil/public/transition/hla/rti/statusboard

Full verification requires passing 1,856 separate tests, each of which involves up to five federates making a series of RTI service calls in a particular order. The tests are designed to ensure that an RTI complies with every statement in the HLA Interface Specification.

Optimizations

The MÄK RTI now allows you to configure its response to the following issues:

- By default, when an instance of the 1516 VariableLengthData class is empty (that is, length is zero), its data method returns a pointer to a single byte initialized to 0. You can change this behavior so that the data method of an empty VariableLengthData instance returns a NULL pointer by enabling the RID parameter **RTI_variableLengthDataUsesNull**.
- You can now use canonical names (host.mak.com) instead of IP addresses in configuration parameters.
- If you experience a race condition when sending interactions using DDM, you can alleviate the problem with the **RTI_dualTransmitFirstInteractionRegions** parameter.
- You can control the degree to which the RTI adheres to name reservation requirements with the **RTI_enableNameReservation** and **RTI_strictNameReservation** RID parameters.
- You can specify the dimensions provided for a conveyed region in the receive interaction service with the **RTI_conveyOnlyAvailableDimensions** RID parameter.
- You can adjust the maximum UDP packet size with the **RTI_maxUdpPacketSize** RID parameter.
- Developers may now more easily override the algorithm used by the RTI's *DtObjectMgr* class to generate object instance handles. Also, you can set the default maximum number of objects per federate with the federation-wide parameter **RTI_maxObjectsPerFederate**.

New Configuration Parameters

MÄK RTI 3.0 has the following new configuration parameters:

- **RTI_variableLengthDataUsesNull** - determines how VariableLengthData represents zero length data. Default: False (to support backwards compatibility with previous versions).
- **RTI_dualTransmitFirstInteractionRegions** - A race condition between a best effort DDM interaction and its region information (sent reliably) is alleviated by transmitting the region information via best effort the first time. Enable this feature to send the DDM interaction region information both best effort and reliably the first time.
- **RTI_enableNameReservation** - The 1516 specification requires name reservation for names used to register object instances. This requirement is can be disabled so that the register object instance service behaves similar to the 1.3 specification (default is enabled).
- **RTI_strictNameReservation** - According to the IEEE 1516 specification, a reserved name cannot be reused in the federation execution. This requirement can be disabled so that a federate may reuse a reserved name once the object has been deleted. The name may be reserved by another federate if the object is deleted and the federate has resigned (default is enabled).
- **RTI_conveyOnlyAvailableDimensions** - Conveyed regions for received interactions contain only the dimensions of the received class, which may be different from the dimensions of the regions used to send the interaction. If this parameter is disabled, then the dimensions of the conveyed regions will match those used to send the interaction.
- **RTI_enableFomBackwardsCompatability** has been renamed **RTI_enableFomBackwardsCompatibility**.
- **RTI_maxUdpPacketSize** – Specifies the maximum size for a UDP packet. This parameter does not affect the underlying UDP configuration. Therefore, the update and interaction messages sent by federates should be restricted to fit within the MTU size. If messages larger than the MTU size are sent, they will be fragmented by the underlying UDP implemenation increasing the chances of packet loss. The default value is 15KB.
- **RTI_reuseReleasedObjectHandles** – Allows the RTI to reuse object instance handles if the original instance has been deleted. Use of this feature breaks the guarantee that handles are unique over the lifetime of a federation. Handles will still be unique at any given instant. This feature is useful in the case of a federate that must register more than **RTI_maxObjectsPerFederate** over the course of a run, but will never have **RTI_maxObjectsPerFederate** at a single instant.
- **RTI_maxObjectsPerFederate** – Allows the federation designer to specify the maximum number of objects that any single federate will register in a single run. Improper use of this parameter will cause the RTI to generate non-unique object handles and will cause undesired and undefined behavior. Most federations should not modify this parameter.

New Example and Example Documentation

This release adds a new example, simpleDDM, which demonstrates how to create a federate that uses DDM. A new appendix has been added to *MÄK RTI Reference Manual*, which describes the rtisimple example and simpleDDM.

Fallback to Lightweight Mode

In previous releases, when the MÄK RTI was configured to use reliable transport (for internal or FOM messages), the RTI would fall back to lightweight mode (no rtiexec and best effort only) if it failed to create a reliable connection. This fall back behavior has been removed and now if the RTI cannot create a reliable connection, it will fail with an RTI internal exception (and pop up message if enabled).

Improved Installation Process on Windows

The MÄK RTI is now added to the beginning of the PATH environment variable so that it will take precedence over any previous installation.

You can install and uninstall the MÄK RTI silently. Instructions for silent installation are on the MÄK Technologies web site, at http://www.mak.com/product_support_faq.php#install and in Chapter 2 in *MÄK RTI Reference Manual*.

The installation package now includes binaries for the example projects.

Documentation Updates

MÄK RTI Reference Manual has been updated for this release. Note the following corrections to the manual:

- In section C.2.2, under the Windows heading, it should read MSVC++ 7.1, not MSVC++ 7.
- In section C.3.2, under the Windows heading, the names of the project files should be *simpleDDMGUI.sln* and *simpleDDMGUI.dsw*.

Bug Fixes

The following bugs have been fixed in this release:

- The spelling of the RID parameter RTI_enableFomBackwardsCompatibility was corrected (from RTI_enableFomBackwardsCompatability).
- The MÄK RTI now correctly handles large messages (greater than 64 KB).

Known Problems

This release has the following known problems:

- 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.
- 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.
- The asynchronous callback feature is not yet available for HLA 1.3 federates. It is only available for 1516 federates. Enabling this option will have no effect on an HLA 1.3 federate.

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.