MÄK Real-time RTI 1.3.4-ngc Release Notes

This document provides the following release-specific information for MÄK Real-time RTI 1.3.4-ngc:

Document ID: RTI-1.3.4-2-010411

Overview

MÄK Real-time RTI 1.3.4-ngc is an implementation of a subset of the HLA Interface Specification, version 1.3.

Although the MÄK Real-time RTI is provided to customers without charge, it requires a license. This license is not necessarily included with the license we provide for products you purchase from us. You can generate a license for the MÄK Real-time RTI on our web site at:
http://www.mak.com/rti_license.htm

Supported Systems

Table 1 lists the platforms supported by the MÄK Real-time RTI 1.3.4-ngc. 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 Real-time RTI libraries.

Table 1: Platforms supported

Platform	Compiler
SGI IRIX6.5	MipsPro7.2.1 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 6.0 (Linux 2.2.16-3)	GNU C++ (g++) distributed with Red Hat distribution. The command: `g++ -v` returns: `gcc version egcs-2.91.66 19990314/Linux (ecgs-1.1.2 release)`
PC with Windows NT 4.0, Windows 95/98/2000	Microsoft Visual C++ 6.0.

Backwards Compatibility

Applications that were built against MÄK Real-time RTI 1.3.x-ngc work with the MÄK Real-time RTI 1.3.4-ngc without recompiling.

Network Compatibility

The MÄK Real-time RTI is not network compatible with the DMSO RTI. (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 one additional function (RtReadFileDescriptor(), described in MÄK Real-time RTI documentation), the MÄK Real-time RTI 1.3.4-ngc has the exact same API as the DMSO RTI 1.3 NG v1, v2, v3.0, and v3.2. Unless you are using this optional function, an application built against the DMSO RTI NG does not need to be recompiled in order to be used with the MÄK Real-time RTI 1.3.4-ngc, or vice versa.

MÄK Real-time RTI 1.3.4-ngc uses the same FED file format as the DMSO RTI, but a different RID file format. (RID file formats depend on the RTI-implementation.)

Changes to Library Names

The library names for MÄK Real-time RTI 1.3.4-ngc match those of the comparable DMSO RTI 1.3 NG-v3 libraries (*libRTI-NG*, *libfedtime*).

On Unix platforms, DMSO RTI 1.3 NG-v3 supplies additional third party libraries (*liborbsvcs*, *libTAO*, *libACE*, and so on). Because these libraries are linked with DMSO's *libRTI-NG*, you should not have to link explicitly with these libraries in any executables that you might build. The MÄK Real-time RTI does not rely on these libraries, and executables provided by MÄK in -ngc releases have been linked ONLY against *libRTI-NG*. This makes them compatible with both the DMSO RTI and the MÄK Real-time RTI.

Federation Time

The MÄK Real-time RTI 1.3.4-ngc files *libfedtime.so* and *libfedtime.dll* implement the same subclass of RTI::FedTime as DMSO's file *libfedtime*. This subclass is called *RTIfedTime* and its definition is in *fedtime.h*. You may use either the MÄK Real-time RTI's *libfedtime* or the DMSO RTI's *libfedtime* with either MÄK's or DMSO's *libRTI*.

Documentation

Because the MÄK Real-time RTI implements the same interface specification as the DMSO RTI, the documentation provided with the DMSO RTI applies to the MÄK Real-time RTI as well, apart from the few differences we describe in this release note and the MÄK Real-time RTI Reference Manual. The DMSO RTI documentation is included with the MÄK Real-time RTI in the *./doc* directory. It is also available on DMSO's HLA web site: http://hla.dmso.mil.

New Features

The MÄK Real-time RTI 1.3.4-ngc has the following new features:

- Ability to choose the network device to use with multicast traffic
- A new RID file option that allows you to choose silent failure over the throwing of exceptions for calls to unimplemented services
- A change to the search-order for RID files, in order to improve compatibility with the DMSO RTI
- New RID file options to control whether or not advisory messages get sent.

Choosing the Network Device to Use

You can choose the network device to use with multicast traffic with the **RTI_mcastDevAddrString** parameter in the *RTI.rid* file. The syntax is:

```
(setqb RTI_mcastDevAddrString "IP_address")
```

where *IP_address* is the IP address of the ethernet device you want to use with multicast traffic.

Choosing Silent Failure Instead of Exceptions

You can have your function calls fail silently rather than throwing an exception when they call a service that is not implemented in the MÄK Real-time RTI. Use the **RTI_useExceptForUnimplServ** parameter in the *RTI.rid* file. The syntax is:

```
(setqb RTI_useExceptForUnimplServ 0)
```

where:

- 0 = do not throw exceptions
- 1 = throw exceptions.

The default is to throw exceptions.

Search Order for RID Files

The MÄK Real-time RTI has changed the search order for RID files to be more compatible with the DMSO RTI. The search order is:

1. The directory and file name specified in the RTI_RID_FILE environment variable, for example, *C:\myconfigdirectory\myrid.mtl*.
2. The *rid.mtl* file located in the directory specified by the RTI_CONFIG environment variable.
3. The *rid.mtl* file located in the directory from which you run the application (which is not necessarily the directory in which the executable is located.)

4. If no RID file is identified by one of the three methods, the MÄK Real-time RTI uses default values.

Controlling Whether or Not Advisory Messages Get Sent

The following new RID file parameters allow you to turn advisory messages for classes, attributes, and interactions, on or off.

```
(setqb RTI_enableClassAdvisory 1)
(setqb RTI_enableAttributeAdvisory 0)
(setqb RTI_enableInteractionAdvisory 1)
```

where:

- 0 = disable advisory messages
- 1 = enable advisory messages.

The default values for sending advisory messages are:

- Classes - enabled (1)
- Attributes - disabled (0)
- Interactions - enabled (1).

Minor Changes and Bug Fixes

This section describes minor changes and bug fixes:

- The federate ambassador turned off class advisories (turnInteractionsOff or stopRegistrationForObjectClass) when the federate unpublished a class, regardless of whether the class advisory had previously been turned off (for example, due to lack of remote federate subscription). As fixed, the RTI does not turn class advisories off when unpublishing a class. The RTI turns class advisories on if the federate republishes the class and the class is relevant (that is, remote federate subscriptions exist). This behavior matches the DMSO RTI.
- The rtiexec exited abnormally when federates repeatedly joined and resigned over a period of time (~100-150 joins). The RTI now safely handles any number of federate joins.
- Class advisories now correctly handle situations where publication and subscription attributes do not overlap. Previously, the RTI could get in a state where class advisories (turnInteractionsOn or startRegistrationForObjectClass) for a published class could not be turned on. This situation occurred when the attributes or parameters in a local publication did not overlap any attributes from a remote subscription. The RTI should not invoke the class advisories under these conditions; however, the RTI entered a state where it failed to invoke advisories even if a subsequent subscription did have overlapping attributes.
- Discovery did not consider ownership properties correctly. A discovery now occurs if and only if the federate is subscribed to a class attribute at the specified class (that is being discovered) and the corresponding object's instance attribute is owned by another federate.
- In some cases, the RTI would not generate discoveries of existing objects for late joining federates. Late joiners will now discover all relevant objects automatically and immediately after subscribing without the subscribing federate having to request class attribute value updates. Reflection of attributes will still require the owning federate to update the object on its own accord, or as a result of the request object attribute value update or the request class attribute value update services.