MÄK Real-time RTI 1.3.5-ngc Release Notes

This document provides the following release-specific information for MÄK Real-time RTI 1.3.5-ngc:

Document ID: RTI-1.3.5-2-010829

Overview

MÄK Real-time RTI 1.3.5-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 1.3.5-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 1.3.5-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 1.3.5-ngc has the exact same API as the DMSO RTI 1.3 NG v1, v2, v3.0, v3.2, and v4.0. 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 1.3.5-ngc, or vice versa.

MÄK Real-time RTI 1.3.5-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 1.3.5-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 and later 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 1.3.5-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

- Updates from unknown objects are no longer processed for discovery.
- The rtiexec requires confirmation before quitting when there are active federations.
- Ability to control the amount of time (in seconds) that the local RTI component waits to receive a response from the rtiexec when processing the createFederation or joinFederation service calls.
- Support for filtering by multicast groups using Declaration Management (DM) services.
- UDP forwarder functionality is available that in some cases can be used to optimize bandwidth usage over a WAN. Please contact MÄK support (support@mak.com) for information about its use.

Processing Discovery of Unknown Objects

By default, the RTI no longer performs discovery processing for updates of unknown objects. A new RID parameter, **RTI_processUnknownUpdatesForDiscovery**, allows you to control whether updates from unknown objects can cause discovery. By default this option is turned off to improve efficiency. Turning it on provides backwards compatibility with RTI 1.3.3 and fault tolerance of dropped register messages when internal messages are sent using UDP. To

```
(setqb RTI_processUnknownUpdatesForDiscovery mode)
```

where *mode* is:

- 0 = do not process unknown updates for discovery
- 1 = process unknown updates for discovery.

Quitting When there are Active Federations

Quitting the RTI Executive while federations are active and federates are still joined, is known to cause abnormal behavior. The RTI Executive now issues a prompt to confirm a quit request when there are active federations. The quit request can be confirmed or canceled. Execution continues while the quit request is processed (that is, processing does not block while waiting for confirmation).

Response Interval Parameter

A new RID parameter, **RTI_responseInterval**, controls the amount of time (in seconds) that the local RTI component waits to receive a response from the rtiexec when processing the createFederation or joinFederation service calls. The default interval is 3.0 seconds.

When using the rtiexec, a federate's local RTI component (LRC) can sometimes experience unusually long delays when communicating with the rtiexec to process the createFederation or joinFederation service calls. As a result, the federate will not be able to complete its connection to the rtiexec reporting "Could not connect to rtiexec" and the service calls will throw an RTIinternalError exception. If a federate repeatedly attempts to join a federation, then a by-product of this problem is the appearance that the federate has joined multiple times in the rtiexec. The solution to this problem is to increase the interval that the LRC will wait for a response.

Other connection issues may cause the LRC to report the same type of connection error (for example, mismatched UDP ports). This error in combination with the appearance of multiple federate joins in the rtiexec is a good indication that the response interval is too small.

The syntax for the new parameter is:

```
(setqb RTI_responseInterval 3.0)
```

Filtering by Multicast Groups Using Declaration Management

The MÄK Real-time RTI has two new RID file parameters, **RTI-addDMObjectMulticastAddr**, and **RTI-addDMInteractionMulticastAddr**, that support the assignment of object and interaction classes to separate multicast groups to allow for basic filtering by multicast groups using Declaration Management (DM) services.

The syntax for the parameters is:

```
(RTI-addDMObjectMulticastAddr "ObjectClassName"
  "MulticastAddress" "NestingOperator")
(RTI-addDMInteractionMulticastAddr "InteractionClassName"
  "MulticastAddress" "NestingOperator")
```

where:

ObjectClassName is a fully qualified FOM class of the form "ObjectRoot.BaseEntity.Platform".

MulticastAddress is a valid multicast address of the form "229.7.7.9".

NestingOperator is a quoted string containing either `exclusive` or `inclusive`. The nesting operator values determine whether the assignment of the multicast address will apply to the specific class exclusively or to that class and all derived classes.

Organize entries in the order of base classes to leaf classes. Exclusive entries must be placed after all inclusive entries from which the class inherits.

In the following example, the address "229.7.7.9" is assigned to all classes derived from "ObjectRoot.BaseEntity.Platform" except for "ObjectRoot.BaseEntity.Platform.GroundVehicle", which is assigned "229.7.7.10":

```
(RTI-addDMObjectMulticastAddr
  "ObjectRoot.BaseEntity.Platform" "229.7.7.9" "inclusive")
(RTI-addDMObjectMulticastAddr
  "ObjectRoot.BaseEntity.Platform.GroundVehicle"
  "229.7.7.10" "exclusive")
```

Using these parameters will result in the following behavior:

- A federate subscription will cause the local RTI component (LRC) to join all multicast groups specified for the subscribed class and its derived classes. For example, given the previous example, a subscription to "ObjectRoot.BaseEntity.Platform" would result in the LRC joining multicast groups "229.7.7.9" and "229.7.7.10".
- The LRC will transmit object attributes to the multicast group specified by the class of the object instance (not the class where the attributes are defined). Given the previous example, the LRC will transmit a “WorldLocation” attribute (defined in class *BaseEntity*) of an object instance of class "ObjectRoot.BaseEntity.Platform.GroundVehicle" to the multicast group "229.7.7.10". It would transmit the same attribute of an object instance of class "ObjectRoot.BaseEntity.Platform" to the multicast group "229.7.7.9".
- The LRC will transmit interactions to the multicast group specified by the interaction class.

The MAK RTI does not implement Data Distribution Management (DDM) services. The functionality described in this section is separate from any filtering that the DDM services might perform.

Documentation Updates

Section 3.9 of MÄK RTI Reference Manual was incorrect. It should read as follows:

3.9 Communicating Over A WAN

The MÄK Real-time RTI supports the specification of multiple destination addresses for best-effort packets. This is often necessary when communicating across a WAN, since most routers do not forward multicast or broadcast packets.

If you are only using reliable transport, you do not need to do anything differently to communicate over a WAN.

In the *rid.mtl* file, use **RTI-addDestAddrString** to add one or more destination addresses. All best-effort data will be sent to each registered address. Configure a separate RID file for each LAN, such that a unicast address for each remote federate (or computer if a computer hosts multiple federates) is added as a destination. The default destination address (destAddrString) is automatically added to the list of destination addresses for LAN best-effort communication, so do not add local federates.

For example, suppose two LANs have the following destination addresses for federates:

- LAN 1:

```
207.86.232.1
207.86.232.2
207.86.232.3
```

- LAN 2:

```
208.86.232.1
208.86.232.2
208.86.232.3
```

Add the following entries to the RID files:

- LAN 1 RID file

```
(RTI-addDestAddrString "208.86.232.1")
(RTI-addDestAddrString "208.86.232.2")
(RTI-addDestAddrString "208.86.232.3")
```

- LAN 2 RID file

```
(RTI-addDestAddrString "207.86.232.1")
(RTI-addDestAddrString "207.86.232.2")
(RTI-addDestAddrString "207.86.232.3")
```

In this situation, you cannot just tell all federates to send to both LANs' broadcast addresses, because broadcast packets will not be forwarded across the WAN.

Minor Changes and Bug Fixes

- Fixed listening to multicast on LAN when using WAN addresses. When using **RTI-addDestAddrString** to specify WAN addresses, the RTI would not listen to the LAN destination address (**RTI_destAddrString**) if this address was multicast. The RTI will now listen to LAN destination addresses that are multicast in conjunction with the use of WAN addresses.

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.