MÄK RTI 2.3.3 Release Notes

This document provides the following release-specific information for MÄK RTI 2.3.3:

Document ID: RTI-2.3.3-2-050113

Supported Systems

Table 1 lists the platforms supported by the MÄK RTI 2.3.3. 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++ 6.0 and 7.1 (.NET)

MÄK RTI 2.3.3 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 2.3.3 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 2.3.3 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 2.3.3 and some use a previous version of the MÄK RTI.) However, the MÄK RTI 2.3.3 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 2.3.3 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 2.3.3, or vice versa (except that the Linux version is not compatible with the DMSO RTI).

MÄK RTI 2.3.3 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 2.3.3, 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 2.3.3 adds Java bindings for HLA 1516.

The MÄK RTI provides a set of Java bindings for both the RTI1.3-NG API and the SISO DLC HLA 1516 API. The Java binding is a thin layer of C++ code that exposes the native C++ API of the RTI to Java applications.

To use the Java bindings, you must install the Java Development Kit (JDK) 1.2 or later and configure your system as described in the following sections.

Configuring Unix Systems to use the Java Bindings

Add or edit the following environment variables:

- Add *makrtix.x/lib/hla.jar* to the CLASSPATH environment variable.
- Linux or Solaris: Add *makrtix.x/lib* to the LD_LIBRARY_PATH variable.
- IRIX: Add the *makrtix.x/lib* to the LD_LIBRARYN32_PATH variable.
- HLA 1516 only: Install the MÄK RTI HLA 1516 Java Fedtime Library as described in "Installing the HLA 1516 Java Fedtime Library," on page 4.

Configuring Windows to use the Java Bindings

Add or edit the following environment variables:

- Add the *makrtix.x\lib\hla.jar* to the CLASSPATH environment variable.
- Add the *makrtix.x\lib* directory to the PATH environment variable.
- (1516 only) Install the MÄK RTI HLA 1516 Java Fedtime Library as described in "Installing the HLA 1516 Java Fedtime Library," on page 4.

Installing the HLA 1516 Java Fedtime Library

HLA 1516 specifies that the federate developer shall provide a Logical Time implementation. For Java federates the MÄK RTI offers two methods of doing this:

- You can create an HLA 1516 time implementation in Java and add the new time class files to the CLASSPATH environment variable.
- You can use the default time implementation provided by the MÄK RTI. The default implements time as a Java double.

Configuring UNIX for the HLA 1516 Java Fedtime Library

Add or edit the following environment variables:

Linux or Solaris

- Add the full path of the directory containing *libjvm.so* to the LD_LIBRARY_PATH variable. This library is distributed as part of the JDK. Its location is system dependent.
- Add the *makrtix.x\lib\java* directory to the LD_LIBRARY_PATH environment variable in front of the *makrtix.x\lib* directory. This ensures that the RTI loads the *libfedtime1516* library required by the Java bindings instead of the default *libfedtime* library.

IRIX

- Add the full path to the directory containing *libjvm.so* to the LD_LIBRARYN32_PATH variable.
- Add the *makrtix.x\lib\java* directory to the LD_LIBRARYN32_PATH environment variable in front of the *makrtix.x\lib* directory. This ensures that the RTI loads the *libfedtime1516* library required by the Java bindings instead of the *default libfedtime* library.

Configuring Windows for the HLA 1516 Java Fedtime Library

Add or edit the following environment variables:

- Add the full path to the directory containing *jvm.dll* to the LD_LIBRARY_PATH variable. This library is distributed as part of the JDK. Its location is system dependent.
- Add the *makrtix.x\lib\java* directory to the PATH environment variable in front of the *makrtix.x\lib* directory. This will ensure that the RTI loads the *libfedtime1516* library required by the Java bindings instead of the default *libfedtime* library.

Developing A Custom Logical Time Implementation

If you are developing a custom Logical Time implementation:

1. Add the custom classes to the CLASSPATH environment variable.
2. Create a new environment variable named RTI_JAVA_TIME_CLASS and set its value to the fully qualified name of the custom *LogicalTimeFactory* class. The fully qualified name is the name of the class preceded by the full package name. The package name must use "/" instead of "." as a separator. For example, the fully qualified name for the default MÄK RTI LogicalTimeFactory is *com/mak/makrti1516/time/DtLogicalTimeDoubleFactory*. In Java, this corresponds to:

```
package com.mak.makrti1516.time;
public class DtLogicalTimeDoubleFactory implements LogicalTimeFactory;
```

3. Create a new environment variable named RTI_JAVA_TIME_INTERVAL_CLASS and set its value to the fully qualified name of the custom *LogicalTimeIntervalFactory* class. The fully qualified name is the name of the class preceded by the full package name. The package name must use "/" instead of "." as a separator. For example, the fully qualified name for the default MÄK RTI LogicalTimeIntervalFactory is: *com/mak/makrti1516/time/DtLogicalTimeDoubleIntervalFactory*. In Java, this corresponds to:

```
package com.mak.makrti1516.time;
public class DtLogicalTimeDoubleIntervalFactory implements
  LogicalTimeIntervalFactory;
```

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:

- This release fixes a memory leak present in previous 2.3.x releases.
- The DDM default ranges are no longer ignored when constructing region realizations.
- 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.

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 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 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.