MAK RTI 4.6 Release Notes

This document provides the following release-specific information for MAK RTI 4.6:

The MAK RTI has been verified by DMSO as fully compliant with the IEEE 1516-2000 version of the HLA Interface Specification (SISO DLC HLA API 1516 (SISO-STD-004.1-2004).)

The US DoD RTI Verification activity was suspended due to budget constraints in January 2015 before declaring any HLA Evolved (IEEE 1516.1-2010) RTIs officially verified. However, at the time that the RTI verification activity was suspended, the MAK RTI had successfully passed 100% of the available tests

Document ID: RTI-4.6-2-200817

Supported Systems

Table 1 lists the platforms supported by the standard version of MAK RTI 4.6. Please contact MAK if you need it for another platform. Application code must be built with the indicated compilers in order to link to the MAK RTI libraries.

Table 1: MAK RTI: Platforms supported

Operating System	Compiler
Red Hat Enterprise Linux Workstation 7 (64-bit libraries) and 8 (64-bit libraries)	Default compiler.
CentOS - versions that are equivalent to supported versions of Red Hat Linux.	Default compiler.
Windows 10	Microsoft Visual C++ 10.0 (32-bit and 64-bit), 12.0 (64-bit), 14.0, 15.0

To use MAK RTI for VC14, you should compile with VS2015 Update 2 or later.

The VC 14 build is binary compatible with all MAK VC 15 products.

The MAK RTI supports Java on all platforms. We build using Java 8, but use Java 6 compliance levels. The MAK RTI is built using the Oracle JDK, but should work with all JDKs. The version of the JVM that you are using must match the bit level of the RTI that you are running (32 or 64).

Compiler Compatibility on Windows

MAK provides versions of product releases that have been compiled with different versions of Microsoft Visual C++. When you run MAK products together on the same computer, for example, the Logger and a VR-Vantage application, we strongly recommend that you run versions compiled with the same compiler. Mixing products compiled with different versions of the compiler on the same computer can result in program instability. The exception to this is that products built with VC 14 and VC 15 are binary compatible.

License Manager

To run the licensed version of MAK RTI, you must install license management software. MAK RTI 4.6 uses FLEXlm 11.13.0.2. If you are upgrading from a version of the MAK RTI that used an older version of FLEXlm, you must upgrade your license management files. You do not need a new license. Licenses are forward compatible.

The License Manager files are not part of the MAK RTI installer. You can download them at:

- Windows:
  http://ftp.mak.com/out/MAKLicenseManager-win-setup.exe (for 32-bit systems)
  http://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- Linux: http://ftp.mak.com/out/MAKLicenseManager-linux-setup.tar.gz (for 32-bit systems)
  http://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

License manager support is available at:

http://www.mak.com/support/licenses

Third Party Library Dependencies

MAK RTI 4.6 has the following dependencies:

- Qt 5.5.1.
- Qwt 6.1.3.
- FreeGlut 2.4.0
- Boost - 1.56 on all platforms except VC 14, which uses 1.63.
- ZLIB 1.2.3 - http://www.zlib.net/
- ICONV 1.8 - http://www.gnu.org/software/libiconv/
- LIBXML2 2.6.22 - http://www.xmlsoft.org/

Backwards Compatibility

The MAK RTI 4.6 libraries are not run-time compatible with previous versions. (In other words, you cannot have a federation execution in which some federates use MAK RTI 4.6 and some use a previous version of the MAK RTI.) However, except as noted below, the MAK RTI 4.6 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*).

HLA Evolved federates compiled with MAK RTI 4.0.4 or later are not link compatible with HLA Evolved federates compiled with MAK RTI 4.0 through 4.0.3.

Network Compatibility

The MAK 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.) 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 Other RTIs

The MAK RTI 1.3 implements the API dictated by the 1.3 version of the HLA Interface Specification, as amended by DMSO's official HLA 1.3 Interpretations document. All RTIs that are verified as compliant with the HLA 1.3 specification are written to this exact API standard. This insures that the MAK RTI is link-compatible with other RTIs that have been verified as HLA compliant.

The MAK RTI 1516-2000 implements the SISO DLC HLA API[1] 1516 (SISO-STD-004.1-2004). This API supports dynamic link compatibility, which was problematic with the original IEEE 1516 API. The IEEE 1516 API only supports compile time compatibility (uses C++ templates and implementation-specific header files). The MAK RTI is compatible with any RTI written to the SISO DLC HLA API 1516.

The MAK RTI HLA Evolved implements the IEEE HLA 1516-2010 API. This API supports dynamic link compatibility, and the MAK RTI is link-compatible with any RTI written to this API specification. There was an error in the HLA Evolved implementation in MAK RTI 4.0, 4.0.1, 4.0.2, and 4.0.3. As a result, those versions are not dynamic link compatible with RTI 4.0.4, RTI 4.1, or any other HLA 1516-2010 compatible RTIs. It is recommended that any HLA Evolved federates compiled against any of those versions of the MAK RTI be recompiled against the latest version.

Because an RTI is typically accessed by an application through a dynamic library (*.dll* on Windows, *.so* on UNIX), applications do not need to be recompiled or relinked to switch among dynamic-link-compatible RTI implementations that have been built using the same compilers. The only requirement is that the appropriate version of the library be located within the shared library search path. For more information, please see Section 3.2, "Compiling and Linking with the MAK RTI 1.3" in *MAK RTI Reference Manual*.

1. Simulation Interoperability Standards Organization Dynamic Link Compatible High Level Architecture Application Program Interface

New Features and Updates

MAK RTI 4.6 has the following new and updated features:

- Support for Red Hat 8. (Please note that Red Hat 6 is no longer supported.)

Documentation Updates

Documentation has been updated for this release. The guides are provided as PDF files in the *./doc* directory. Online help is available from the MAK RTI Help menu. You can also find the most current PDFs at https://www.mak.com/support/product-user-guides.

Bug Fixes

The following bugs (with defect tracking numbers) have been fixed in this release:

- When a federate publishes intermediate classes as well as the leaf class, an object registered at the leaf class will be discovered at the leaf class (if subscribed), not the intermediate class. 64792 / RTI-418
- The FFC setting is now preserved and is consistent when stopping and restarting rtiexec through the assistant. 66505 / RTI-435, RTI-463
- Lightweight federates from remote machines display in federation view as intended. 66542 / RTI-437, RTI-464

Known Problems

This release has the following known problems:

- All RTI Assistants that expect to communicate with each other must use the same port.
- The rtiForwarder may crash when using multiple rtiForwarders and when using routers, VPNs, or both, if the *rid.mtl* file specifies the same value for the RTI_udpPort and RTI_tcpPort parameters. 54545
- If multicast is disabled on a network, the RTI Assistant network map and federations table may not display properly. This problem does not affect the functioning of the RTI, just the display in the RTI Assistant.
- The RTI Assistant's RTIspy window can display network and performance statistics for any federate in your federation that has RTI_enableLRCNetworkStatisticsMonitoring enabled in its RID file. This monitoring capability is implemented by a plugin to the LRC. The RTIspy can only have one plug-in loaded at a time. Therefore, the RTIspy window will not work if you are using your own LRC plugin.
- Shared memory does not work with 64-bit Linux federates.
- Some computers running Windows Vista may experience problems receiving IPv6 multicast messages. This can affect communication between RTI Assistants, which makes it impossible to detect active connections on your network, and best-effort communications between federates.
- Distributed forwarder multicast subscription does not work for late-joining forwarders. With the introduction of the unified Distributed Forwarder a need arose to communicate between federates on a LAN and forwarders on remote LANs when a DM or DDM multicast group is joined or dropped, such that the forwarders can monitor or drop those groups on its LAN and forward messages back to the federates joined to such multicast groups. However the scheme to accomplish this does not work if the forwarder is not connected to the forwarder network when the federate joins the multicast group. The workaround is to ensure that the forwarder network is completely established prior to launching federates, or at the least, prior to allowing federates to attempt to join DM or DDM multicast groups.
- The MAK Technologies Lisp (MTL) files used by MAK products to specify configuration settings are parsed at startup by the RTI. When a configuration file has certain errors, such as unmatched parentheses, the parser will not necessarily return an error code. This can cause the RTI to reach a partially initialized state or cause the RTI to initialize itself with default RID values, causing unexpected behavior. This applies to lisp expressions in the RID file as well as the expressions passed to the 1516 createRTIambassador function.
- A workaround has been added to deal with situations where a host running the RTI Forwarder application has multiple network interface adapters (NICs) or has multiple IP addresses assigned to an interface, which may be the case if the RTI Forwarder host is connected to the WAN via a VPN tunnel. Under these circumstances the RTI Forwarder may fail to initiate connections to other forwarders if the IP address corresponding to its host in the Forwarder List section of the *routes.mtl* file does not match the IP

address the RTI Forwarder application receives when it queries the host itself for its IP address. To correct this the IP addresses of the interfaces on which it will connect to other RTI Forwarders must be entered into the *routes.mtl* file Interface Address List section. At a minimum one such entry should correspond to the entry identifying the host in the Forwarder List section of the *routes.mtl* file. Please refer to Section 10.6, "Configuring RTI Forwarders for Distributed Forwarding" in *MAK RTI Reference Manual* for details about the Interface Address List entry format.

- 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.