MÄK TECHNOLOGIES

RTI

The Run-Time Infrastructure for HLA

MäK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138 USA

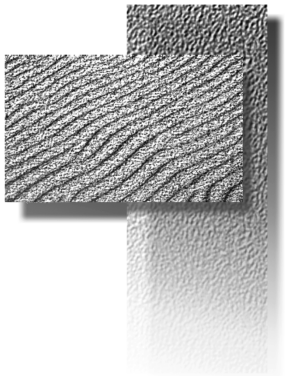
MÄK Technologies
185 Alewife Brook Parkway
Cambridge, MA 02138 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com

www.mak.com

Revision RTI-1.0.3beta-1-980611



Contents

The MÄK Runtime Infrastructure

Preface

This manual supports the Beta release of the MäK RTI 1.0.3. The MÄK RTI is an implementation of the Run-Time Infrastructure portion of the High-Level Architecture (HLA). The MÄK RTI 1.0.3 implements a subset of the HLA Interface Specification version 1.1.

About this Manual

This manual is for developers of HLA simulation applications and for persons who need to install the MäK RTI.

This manual assumes that you are familiar with basic UNIX or DOS operations and that you know how to work in your computer environment, either OSF/Motif or Windows.

The MäK RTI is a dynamic product, constantly evolving to meet user needs. Consequently, you may find improvements or new features that have been added to the RTI since this manual went to press.

Other MÄK Products for the Virtual World

The RTI is a member of MÄK Technologies' line of software products designed to streamline the process of developing and using networked simulated environments.

In addition to the RTI, the MÄK Technologies product line includes:

- The VR-Link Network Toolkit. The Toolkit is an object oriented library of C++ functions and definitions that implement the HLA RPR FOM and the DIS protocol. This library minimizes the time and effort required to build and maintain new HLA/DIS-compliant applications, and to integrate such compliance into existing applications.

  The Toolkit includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.

  VR-Link is the basis of the other MÄK products.

- The MÄK Stealth. The Stealth displays a realistic, three-dimensional view of your virtual world. You can view this world from the inside of a simulated moving vehicle, or place the eyepoint at another moving or stationary location. The Stealth lets you switch rapidly among several predefined viewpoints while the simulation is underway.

- The MÄK Logger. The *Logger* , ltl ,c- r-6.4(e... -3.4(H)7.9
  et atss6095phedss6095 ab65(e L)-5.9(r-4.9(b

Getting Technical Support

For RTI technical support, information about upgrades, and information about other MÄK products, you can reach us in the following ways:

For sales and upgrade information by e-mail:	vrlink-info-sales@mak.com
For technical support by e-mail:	vrlink-tech-spt@mak.com
For general MÄK information via the Web:	http://www.mak.com

Send postal correspondence to:	MÄK Technologies 185 Alewife Brook Parkway Cambridge, MA 02138

Call or fax us at:	Fax:	617-876-9208
	Voice Phone:	617-876-8085

Document Conventions

This document uses the following typographic conventions:

`Monospaced Type`	indicates examples of code or commands, or values you enter as written.
`Monospaced Italic`	is used in command examples to indicate variables that you replace with appropriate values.
`[ ]`	indicates optional arguments.
`|`	separates options in a command where only one option may be chosen at a time.
Italic	indicates a file name or path, or a class name.
>	indicates a one-step procedure.
Menu→Option	Menu options appear as commands linked together by an arrow. For example, an instruction to select the Save option from the File menu appears as: Choose File→Save.

Directory names are preceded with dot and slant characters that show their position with respect to the RTI home directory. For example, the directory *makrti/bin* appears in the text as *./bin.*

The MÄK Runtime Infrastructure

The MÄK RTI is an implementation of the Run-Time Infrastructure portion of the High-Level Architecture (HLA). The MÄK RTI 1.0.3 implements a subset of the HLA Interface Specification version 1.1.

MÄK RTI 1.0.3 is the initial release of the MÄK RTI. It was given the revision number 1.0.3, since this is the revision number of the DMSO RTI with which it is link-compatible.

System Requirements

The MÄK RTI 1.0.3 beta supports the following platforms (other platforms will follow):

- SGI IRIX6.2 or later (both n32 and o32ABIs): Requires MipsPro7.1 C++ compiler (available from SGI). No particular patches are required.
- Sun Sparcstation with SunOS4.1.3 or later: Requires GNU C++ (g++) version 2.7.2.3 and G++ libraries version 2.7.2. (Available via anonymous FTP from prep.ai.mit.edu/pub/gnu.
- Sun Sparcstation with Solaris2.5 or later: Requires SPARCompiler C++ version 4.1 (available from Sun).
- Dec Alpha with OSF/1 V4.0: Requires DEC C++ for OSF/1 version 5.5 (available from DEC).
- PCs with Windows NT or Windows 95: Requires Microsoft Visual C++ 4.2 or later.

Compatibility with the DMSO RTI

With the exception of one additional function (see "Accessing The RTI's File Descriptor" on page 10), the MÄK RTI has the exact same API as the DMSO RTI. The MäK RTI also uses the exact same header files as those supplied with DMSO RTI 1.0.3. This section discusses the following aspects of compatibility:

- Compiling
- Linking
- FED file
- RID file
- Network

Compiling

An application built against the DMSO RTI does not need to be recompiled in order to be used with the MÄK RTI, or vice versa. For more information, see "Compiling and Linking" on page 3.

Dynamic Linking

Because the RTI is typically accessed by an application through a dynamic library (called *libRTI.so* on UNIX platforms and *rti.dll* on PCs), applications do not need to be relinked in order to switch between the MÄK RTI and the DMSO RTI. Applications built using the DMSO RTI should run with the MÄK RTI unmodified. All that is required is that the appropriate version of *libRTI.so* or *rti.dll* be located within the shared library search path. For more information, see "Compiling and Linking" on page 3.

Static Linking

The MÄK RTI contains a static version of the library (*libRTI.a*). If you link statically with this library, you must relink to switch to the DMSO RTI.

FED File

The MäK RTI uses the same FED file format as the DMSO RTI. (The FED file is an RTI-readable description of the relevant information in the FOM). It is not necessary to modify your FOM or FED file to switch among the two. See "The FED File" on page 9 for details.

RID File

The MäK RTI uses a different RID file format than the DMSO RTI. The RID (RTI-Initialization Data) file provides RTI-implementation-specific configuration parameters, and hence cannot be shared among different RTIs. A MäK RTI RID file is an *.mtl* file with the same format as the files used to configure other MÄK products. See "The RID File" on page 8 for details.

Network Compatibility

The MÄK RTI is not network compatible with the DMSO RTI. (In general, any two different RTI implementations will not be network compatible.) This means that if a set of applications want to play together in the same federation execution, they must all be using the same RTI implementation. (But, of course, they can all then switch to another RTI implementation and play in another federation execution).

Documentation

Because the MÄK RTI implements the same interface specification as the DMSO RTI, the documentation provided with the DMSO RTI applies to the MÄK RTI as well, apart from the few differences we describe below. The *DMSO RTI Programmer's Guide* can be found on DMSO's HLA web site: http://hla.dmso.mil. Choose HLA Software Distribution Center, then click the Products button. Scroll down to any of the RTI 1.0.3 distributions, and download the *Programmer's Guide*.

Compiling and Linking

To build an application that uses the MÄK RTI, you must specify the RTI's include directory on your compile line, and the RTI's *lib* directory and the name of the RTI library on your link line.

For example, if you have installed the RTI in */usr/products/makRti1.0.3*, the following should appear on your compile line:

```
-I/usr/products/makRti1.0.3/include
```

and the following should appear on your link line:

```
-L/usr/products/makRti1.0.3/lib -lRTI
```

The MÄK RTI uses *libm*, the standard system math library. Therefore the flag `-lm` must appear on your link line after -lRTI.

In addition, under Solaris, you will need the following flags on your link line:

- `-lsocket`
- `-lnsl`
- `-lint`

because the MÄK RTI uses these system libraries.

Running Applications

If you have linked dynamically with the *libRTI.so* or *rti.dll* library, the library needs to be accessible to the run-time linker, like any other shared library.

> To make the library accessible to the run-time linker, include the RTI's *lib* directory in the dynamic library search path, which is configurable through an environment variable on most systems.

Table 1 lists the envrionment variables for the systems supported by the MäK RTI.

Table 1: Environment variables

System	Environment Variable
IRIX6 N32 ABI	LD_LIBRARYN32_PATH
IRIX5 O32 ABI	LD_LIBRARY_PATH
SunOS4	LD_LIBRARY_PATH
Solaris	LD_LIBRARY_PATH
Dec Alpha	PATH
Win NT/95	PATH

As an alternative, you can copy the library to the standard place that shared libraries are stored on your system (typically in */usr/lib* on UNIX machines, or in *\winnt\system32* or *\windows\system* on PCs).

License Manager

Like all other MÄK products, the MÄK RTI uses the FLEXlm license manager. We have tried to set things up so that this has minimal impact on users, but there are a few steps you need to take before running MÄK-RTI-based applications.

Setting up the License Server

Choose a machine to be your MÄK license server. After installing the MÄK RTI (or other MÄK product) on that machine, perform these steps. These steps need only be done once when you obtain a new MÄK product.

1. While logged onto the server, switch to the *flexlm* directory under the MÄK RTI home directory and execute *lmhostid*. This will return a unique number that MÄK will use to generate the activation codes for your license.
2. Call or email MÄK (keys@mak.com) with the name of the server and the number obtained by *lmhostid*. MÄK will email, FTP or FAX you a file called *license.dat*.
3. Put *license.dat* in the *flexlm* directory under the MÄK RTI home directory on all machines on which you have the MÄK RTI installed, including the server.
4. If you are running Windows NT, edit the *license.dat* file to specify the location of the *maklmgrd* executable. Initially, the file contains a dot to represent this location, as in the first line below. Replace the dot with the absolute path to *maklmgrd*, as in the second line:

```
DAEMON maklmgrd .
DAEMON maklmgrd C:\MAKRTI1.0.3\flexlm\maklmgrd
```

Running the License Server

The FLEXlm license server needs to be running on the server machine before you can run any MÄK-RTI-based applications. Prior to running the server, set the environment variable LM_LICENSE_FILE to the absolute path to *license.dat*. For example, under UNIX, the environment command might look like this:

```
setenv LM_LICENSE_FILE /usr/local/makRti1.0.3/flexlm/license.dat
```

or under Windows NT, like this:

```
set LM_LICENSE_FILE=C:\makrti1.0.3\flexlm\license.dat
```

> To start the FLEXlm license server, execute *runLm* from the MÄK RTI's *flexlm* directory.

> To obtain the current status of the server, run *lmstat*.

> To force all applications to check in their licenses, and then shut down the server, run *lmdown*.

Running Applications

Before you run any MÄK-RTI-based applications, the license server must be running. In addition, you must set the LM_LICENSE_FILE environment variable, as described in "Running the License Server", on each machine on which you are running a MÄK-RTI-based application.

Services Implemented

MÄK RTI 1.0.3 implements a subset of the RTI services. Specifically, we have implemented the functionality that is most likely to be needed by the real-time platform-level simulation community.

The following federate-initiated services are implemented (stubs exist for the rest):

- CreateFederateExecution()
- DestroyFederateExecution()
- JoinFederateExecution()
- ResignFederateExecution()
- PublishObjectClass()
- PublishInteractionClass()
- SubscribeObjectClassAttribute()
- SubscribeInteractionClass()
- RequestID()
- RegisterObject()
- UpdateAttributeValues()
- SendInteraction()
- DeleteObject()
- RequestAttributeValueUpdate()
- Tick()

The following RTI-initiated services can be called by the MÄK RTI:

- StartInteractionGeneration()
- StopInteractionGeneration()
- StartUpdates()
- StopUpdates()
- DiscoverObject()
- ReflectAttributeValues()
- ReceiveInteraction()
- RemoveObject()
- ProvideAttributeValueUpdate()

The MÄK RTI 1.0.3 only implements the best-effort communications mechanism, and does not publish any MOM data.

The Optional rtiexec

No server or other central application is necessary in order to use the MÄK RTI. Just start the applications, and they should be able to communicate, as long as they are all configured to use the same multicast or broadcast address and UDP port (see "The RID File" on page 8).

However, you may choose to use an optional *rtiexec* application to keep track of the current set of valid federation executions, and to guarantee that federate IDs assigned to federates joining a federation execution are unique.

If you are not using the *rtiexec*:

- The createFederationExecution() service returns successfully even if the named execution has already been created
- The destroyFederationExecution() service always returns successfully even if the named execution does not currently exist
- The joinFederationExecution() service returns successfully even if the named federation execution has never been created using createFederationExecution()

In addition, the federate ID returned by the joinFederationExecution() service is not guaranteed to be unique when not using the *rtiexec*. It is computed by taking the federate's process ID modulo 10,000. In an execution with ten federates running on ten different machines, there is about a one in 200 chance that the two federates will be assigned the same federate ID.

> To use the *rtiexec*, run *rtiexec* from the MÄK RTI's *bin* directory, making sure that it is configured to use the same broadcast or multicast address and port number as all of the federates. (See "The RID File" on page 8).

> To override port number in the *RID* file use *rtiexec*'s `-P` option.

> To override the IP address use the `-A` option.

For example:

```
rtiexec -P 4444 -A 225.9.9.9
```

You must indicate to the federates that they should try to connect to the *rtiexec.*

> To indicate to the federates that they should try to connect to the *rtiexec,* set the **RTI_useRtiExec** parameter to 1 in the RID file.

If you are using the *rtiexec* in a particular federation execution, it should remain running until that federation execution has been destroyed.

> To exit the *rtiexec*, type "q" and the return, in its console.

> To inhibit the output of diagnostics, use the `-q` option.

This is often desirable when running *rtiexec* in the background.

The RID File

When an application creates an instance of *RTI::RTIambassador()*, the RTI can load a RID file that contains various configuration parameters. This file is called *rid.mtl*. The RTI looks for this file first in the current directory (./), then in the directory indicated by the environment variable RTI_CONFIG. If no *rid.mtl* file is found, or if some parameters do not appear in the file, default parameter values are used.

A MÄK RTI RID file is an *.mtl* file of the same form as the files used to configure other MÄK products. It contains a line for each parameter that you would like to set, in the format:

```
(setqb parameterName value)
```

Table 2 lists the supported parameters.

Table 2: RID file parameters

Parameter	Default	Description
RTI_useRtiExec	0 (do not use *rtiexec*)	Determines whether the application should attempt to connect to an *rtiexec* application when it creates, destroys or joins a federation execution. Valid values are 0 and 1.
RTI_udpPort	4000	Indicates the UDP port number to be used for messages sent among federates, as well as between federates and the *rtiexec* if applicable. All federates in a federation execution must use the same value.
RTI_destAddrString	229.7.7.7 (a multicast address)	Indicates the IP address to which federation execution traffic is sent. This is typically a broadcast address or multicast address. All federates in a federation execution must use the same value. A value of "0.0.0.0" is a special value that indicates that the *default _broadcast_ address* should be used. If any of your federates are running on platforms on which multicast is not supported, you must use a broadcast address rather than a multicast address.
RTI_checkFlag	1	Indicates whether the RTI should check the validity of attributes and parameters and the ownership of attributes passed to updateAttributeValues() and sendInteraction(). Setting this flag to 0 results in a small performance gain, but it means that the AttributeNotDefined, AttributeNotOwned, and InteractionParameterNotDefined exceptions will never be thrown by these services. Valid values are 0 and 1.

A sample *rid.mtl* file is provided in the MÄK RTI root directory.

The FED File

The MÄK RTI uses the same FED file format as the DMSO RTI. When the joinFederationExecution() service is invoked, the RTI reads the FED file named *execName.fed*, where *execName* is the name passed to the join call. The MäK RTI first looks in the current directory (./), and then in the directory indicated by the RTI_CONFIG environment variable. All federates in a federation execution must use identical FED files.

Accessing The RTI's File Descriptor

The MäK RTI adds one function to the API provided by the DMSO RTI. The function is called RtReadFileDescriptor() and is declared in *makRti.h*. This function returns a file descriptor on which data is guaranteed to be present whenever there is input from the network that the RTI needs to process.

This file descriptor can be passed to the system call select (or to something like the X function XtAppAddInput()) in applications that wish to yield the CPU until there is HLA input pending, avoiding unnecessary polling *using RTI::tick()*. When the application regains control, *RTI::tick()* should be called so that the pending input can be processed by the RTI.

RtReadFileDescriptor() has the following prototype:

```
int RtReadFileDescriptor(RTI::RTIambassador* amb);
```

Note: If you use this extension to the standard RTI API, your application will no longer be link-compatible with the DMSO RTI, as it does not implement this function. You can place the calls within #ifdefs, if your application does need to switch between RTI implementations.

rtidump

The MÄK RTI includes a debugging utility called *rtidump*, that can passively listen to a federation execution and print out all messages being exchanged by the RTI on the network.

rtidump must be configured to be listening to the port and address being used for the desired federation execution.

> To configure rtidump to listen to the port and address being used for the desired federation execution, use *rid.mtl*, or the `-P` and `-A` command-line options.

Command line options take precedence over the *.mtl* values.

> To quit *rtidump*, press "q", then return.

Packet Server (UNIX Only)

On UNIX systems, you can off-load the MÄK RTI's reading of packets from the network to its own asynchronous process using the MÄK Packet Server - a separate product available from MÄK. This is particularly useful on multiprocessor machines, and in cases where you're running multiple federates on the same machine. In the latter case, a packet would be read from the network only once by the packet server, and viewed by each federate through shared memory.