

# HLA RTI 1.3 Next Generation Implementation

*RTI-NG 1.3v3.2 Installation Guide*

Copyright © 1999-2000 Science Applications International Corp.,  
All Rights Reserved

The Next Generation RTI computer software and documentation has been developed under contract N61339-97-C-0073 in which the regulations from DFARS 252.227-7013 and 252.227-7014 are enforced. This material may be reproduced by or for the U.S. Government pursuant to the copyright license under these DFARS clauses.

Science Applications International Corporation  
Technology Research Group  
Distributed Computing Technology Division  
5400 Shawnee Road, Suite 110  
Alexandria, VA 22312

<b>1</b>	<b>INTRODUCTION .....</b>	<b>7</b>
1.1	ABOUT THIS MANUAL .....	7
1.2	DOCUMENTATION .....	7
1.3	CUSTOMER SUPPORT.....	8
1.3.1	<i>Web Based Helpdesk System.....</i>	<i>8</i>
<b>2</b>	<b>INSTALLING THE RTI .....</b>	<b>9</b>
2.1	PLANNING THE INSTALLATION .....	9
2.1.1	<i>Supported Configurations .....</i>	<i>9</i>
2.1.2	<i>System Requirements.....</i>	<i>10</i>
2.2	RTI DIRECTORY STRUCTURE.....	10
<b>3</b>	<b>INSTALLING THE RTI ON UNIX PLATFORMS .....</b>	<b>12</b>
3.1	CREATE THE INSTALLATION DIRECTORY .....	12
3.2	DOWNLOAD THE RTI DISTRIBUTION .....	12
3.3	INSTALLING THE DISTRIBUTION .....	13
3.3.1	<i>Troubleshooting Installation Problems.....</i>	<i>14</i>
<b>4</b>	<b>INSTALLING THE RTI ON WINDOWS 98/NT/2000 .....</b>	<b>15</b>
4.1	CREATE THE INSTALLATION FOLDER .....	15
4.2	DOWNLOAD THE RTI DISTRIBUTION .....	15
4.3	INSTALLING THE DISTRIBUTION .....	15
<b>5</b>	<b>INSTALLING THE RTI ON A UNIX PLATFORM FOR VXWORKS.....</b>	<b>16</b>
5.1	DOWNLOAD AND INSTALL THE RTI DISTRIBUTION .....	16
5.2	PREPARE A VXWORKS KERNEL FOR RTI OPERATION .....	16
5.2.1	<i>Configure Kernel Options.....</i>	<i>16</i>
5.2.2	<i>Add Green Hills Compiler Support to the Kernel.....</i>	<i>16</i>

5.2.3	<i>Increase the Task Stack Size</i> .....	17
5.2.4	<i>Enable Kernel Support for UDP Multicast (Optional)</i> .....	17
5.3	MISCELLANEOUS VXWORKS INSTALLATION/DEVELOPMENT NOTES .....	18
5.3.1	<i>RTI Stack Objects</i> .....	19
5.3.2	<i>Slow Application Load Times</i> .....	19
5.3.3	<i>System Clock Rate and Tick</i> .....	19
<b>6</b>	<b>TESTING THE RTI ON UNIX PLATFORMS .....</b>	<b>20</b>
6.1	ENVIRONMENT CONFIGURATION .....	20
6.2	BUILD THE HELLOWORLD APPLICATION.....	21
6.3	EXECUTE HELLOWORLD .....	22
<b>7</b>	<b>TESTING THE RTI ON WINDOWS NT.....</b>	<b>24</b>
7.1	ENVIRONMENT CONFIGURATION .....	24
7.2	BUILDING THE HELLOWORLD APPLICATION .....	25
7.3	EXECUTE HELLOWORLD .....	25
<b>8</b>	<b>TESTING THE RTI ON WINDOWS 98.....</b>	<b>27</b>
8.1	ENVIRONMENT CONFIGURATION .....	27
8.2	BUILDING THE HELLOWORLD APPLICATION .....	28
8.3	EXECUTE HELLOWORLD .....	28
<b>9</b>	<b>TESTING THE RTI ON VXWORKS.....</b>	<b>30</b>
9.1	ENVIRONMENT CONFIGURATION .....	30
9.2	BUILDING THE HELLOWORLD APPLICATION .....	30
9.3	EXECUTE HELLOWORLD .....	32
<b>10</b>	<b>RUNNING APPLICATIONS WITH THE RTI.....</b>	<b>34</b>
10.1	RTI-NG COMPONENTS .....	34
10.2	RTI CONFIGURATION FILES.....	34

10.2.1	<i>FED File</i> .....	35
10.2.2	<i>RID File</i> .....	35
10.2.3	<i>RID Consistency</i> .....	36
10.3	RTI COMMUNICATIONS OVERVIEW .....	37
10.4	THE RTI CONSOLE APPLICATION .....	39
10.4.1	<i>Usage</i> .....	39
10.4.2	<i>Commands</i> .....	40
10.4.3	<i>Notes</i> .....	43
10.5	RTI CONFIGURATION TOOL.....	43
10.5.1	<i>Usage</i> .....	44
10.5.2	<i>Notes</i> .....	45
<b>APPENDIX A. PLATFORM SPECIFIC INFORMATION .....</b>		<b>46</b>
10.6	SOLARIS 2.6 .....	46
10.7	REQUIRED SYSTEM PATCHES FOR SOLARIS 2.6 PLATFORMS.....	46
10.8	SOLARIS 2.7 .....	47
10.9	REQUIRED SYSTEM PATCHES FOR SOLARIS 2.7 PLATFORMS.....	47
10.10	IRIX 6.5 .....	48
10.11	LINUX 2.2 .....	48
10.12	REQUIRED SYSTEM PATCHES FOR LINUX 2.2 PLATFORMS.....	49
10.13	WINDOWS NT 4.0.....	49



## 1 Introduction

The Run-Time Infrastructure Next Generation 1.3 (RTI-NG 1.3) is an implementation that corresponds to the High Level Architecture Interface Specification version 1.3 (<http://hla.dmsso.mil/tech/ifspec.html>). The RTI provides a collection of common services used to support the modeling and simulation community. All of these services are accessed through a standard programming language API. The C++, Java, and Ada 95 programming languages are supported, as well as the CORBA Interface Definition Language. Information concerning the HLA and RTI can be found on the DMSO (Defense Modeling and Simulation Office) web site, <http://hla.dmsso.mil>.

The RTI-NG 1.3 implementation has been developed under the sponsorship of the Defense Modeling and Simulation Office and contracted through the US Army Simulation, Training and Instrumentation Command (STRICOM). The development team has been led by the Sciences Application International Corporation (SAIC) Technology Research Group and supported by Virtual Technology Corporation, Object Sciences Corporation, Dynamic Animation Systems, Washington University in St. Louis, and University of California at Irvine.

### 1.1 About This Manual

This manual is for HLA federate developers and systems administrators who will be installing the RTI-NG software. Installation begins with proper preparation to ensure that the host platform is compatible with the RTI software requirements for proper operation. The process should be relatively automatic in which a script will attempt to extract the contents of the distribution package and copy onto the appropriate location of the host platform. The details of the installation process may be dependent on the particular platform (i.e., combination of machine architecture, operating system, and compiler) and this document is organized accordingly.

After the installation is complete the software needs to be tested to ensure proper RTI installation and operation. This manual contains the necessary instructions to build a simple HLA federate application used to exercise the basic RTI services. The source code and supporting make (or project) files for the example *helloWorld* application are provided with the RTI distribution. Instances of the *helloWorld* application can be executed on multiple machines in order to demonstrate proper RTI connectivity across the network.

### 1.2 Documentation

The RTI-NG documents are listed below. These documents can be downloaded individually from the DMSO Software Distribution Center (<http://hla.dmsso.mil/sdc>). This documentation set includes:

**Installation Guide** – This document explains how to install the RTI-NG software and verify that the installation is working properly. The installation verification process involves the building and execution of a simple federate application, helloWorld, which exercises a number of RTI services and exchanges information between two or more federates.

**Release Notes** – This document contains any known issues for the latest release of the RTI-NG software. The known issues may include potential installation problems, software defects, or problems with adherence to the HLA Interface Specification.

**Programmer's Guide** – This document describes how to develop federate applications using the RTI-NG software. Included is a description of all the RTI services and federate callbacks with additional guidance based on the behavior of the RTI-NG implementation.

## **1.3 Customer Support**

### **1.3.1 Web Based Helpdesk System**

A web based helpdesk system has been established (<http://helpdesk.dctd.saic.com>) to allow users of the DMSO RTI software to submit problem reports or enhancement requests. The helpdesk system automates the collection of information related to problem reports and supports the necessary workflow to ensure that issues are resolved in a timely manner.

Each user of the helpdesk system is required to have a valid account with the DMSO Software Distribution Center (SDC). The SDC is responsible for supporting downloads of the DMSO HLA RTI and tools. Any existing SDC users have already been registered with the RTI helpdesk system using the same account name and password. The registration form for the SDC can be found at <http://hla.dmsso.mil/sdc>.

The helpdesk web pages contain information concerning the operation of the support system. The helpdesk system provides mechanisms to submit problem reports, view the status of a user's reports, view solutions to a particular problem report, and search for common problems and corresponding solutions.

## 2 Installing The RTI

### 2.1 Planning The Installation

Prior to installing the RTI, you should take a few minutes to insure that your system is properly configured to support an RTI installation and operation. This includes checking to insure there is adequate disk space and that the appropriate system and compiler patches have been installed. In addition, it is prudent to check the downloaded file sizes against the posted sizes on the SDC page.

#### 2.1.1 Supported Configurations

Table 1 lists the platforms and compilers currently supported by RTI-NG.

Table 1. Supported Platforms

Operating System	Compiler Type	Build Type
SunOS 5.7 (SPARC)	GCC 2.95.2	Optimized, Multithreaded
SunOS 5.7 (SPARC)	SunPro 5.0	Optimized, Multithreaded
SunOS 5.6 (SPARC)	SunPro 4.2	Optimized, Multithreaded
RedHat Linux 6.1 (Intel)	EGCS 1.1.2	Optimized, Multithreaded
Irix 6.5.6 (MIPS 4)	MIPSpro 7.3.1.1m	Optimized, Multithreaded
Irix 6.5.3 (MIPS 3)	MIPSpro 7.2.1.3m	Optimized, Multithreaded
VxWorks 5.3.1 (PowerPC)	GreenHills 1.8.9	Optimized, Multithreaded
Windows 98, Second Edition	MS Visual C++ 6.0 SP3	Optimized, Multithreaded
Windows NT 4.0 SP5	MS Visual C++ 6.0 SP3	Optimized, Multithreaded

## 2.1.2 System Requirements

The RTI-NG 1.3 Release Notes document contains information about system hardware requirements, as well as any required patches that may be needed to insure proper operation of the software. Please consult this document and resolve any system deficiencies prior to attempting installation.

Unix distributions require the GNU make (gmake) utility to compile the sample applications provided with the libraries. This package is included in binary form with the RTI distribution. It is also freely available from <ftp://ftp.gnu.org/pub/gnu/make>, as well as other sites on the Internet. Only gmake version 3.76 or greater is supported. To check the version of your gmake installation, type

```
rtiuser % gmake -v
```

```
GNU Make version 3.76.1, by Richard Stallman and Roland McGrath.  
Copyright (C) 1988, 89, 90, 91, 92, 93, 94, 95, 96, 97  
Free Software Foundation, Inc.
```

## 2.2 RTI Directory Structure

The RTI-NG 1.3 directory structure after installation is depicted in Figure 1. This layout permits the installation of multiple build types of the RTI, as well as multiple versions in the same installation area.

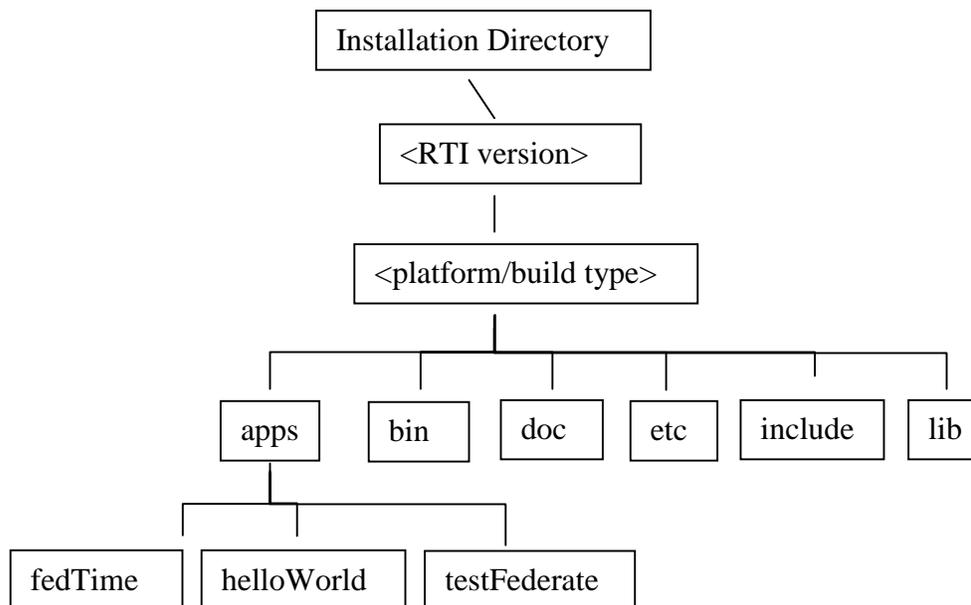


Figure 1 - RTI Installation Directory Structure

A description of the contents of each directory is provided in Table 2.

Table 2 - Installation Directory Contents

<b>Directory</b>	<b>Contents</b>
Installation Directory	One or more RTI-NG 1.3 versions
<rti version>	One or more RTI-NG 1.3 build types
<platform/build type>	RTI-NG 1.3 binaries and sample source for a single platform/ build type
apps	RTI sample applications
bin	RTI executables – rtiexec, fedex, launch
doc	RTI documentation suite, sample fed and RID files
etc	RTI configuration tool and supporting files
include	HLA C++ API headers
lib	RTI and supporting libraries

## 3 Installing the RTI on Unix Platforms

### 3.1 Create the installation directory

The first step is to create an installation directory. There should be sufficient space on the disk where this directory is created to permit installation of the RTI binaries, and for building sample applications. The DMSO Software Distribution Center provides information on disk space requirements for each distribution package.

The directory can be any name you choose. If a single installation of the RTI is to be shared across multiple hosts, it is recommended that this installation directory be exported to the other hosts using NFS or similar network-based file system. The RTI-NG installation directory structure supports the coexistence of multiple distributions for different platforms. Configuring this area for networked file systems is beyond the scope of this manual, but these procedures should be available in your system documentation.

Note that in the command lines below, the user's prompt looks like this:

```
rtiuser %
```

The commands to be typed by the user are in boldface.

To create the directory, type:

```
rtiuser % cd /home/data1  
rtiuser % mkdir rti
```

Make sure that this directory is writable:

```
rtiuser % ls -l  
total 6  
drwxr-xr-x  3 testsol  sim          512 Sep 27 17:40 rti/
```

If the directory is not writable, change permissions as follows:

```
rtiuser % chmod u+w rti
```

Then change directory to this directory:

```
rtiuser % cd rti
```

### 3.2 Download the RTI Distribution

The RTI-NG 1.3 distribution is freely available online from the DMSO Software Distribution Center (SDC). For Unix platforms, the RTI is distributed in shell archive format.

When it is executed, a shell archive (simply a unix shell script) will uncompress, install and configure its contents.

The distribution file should be placed in the installation directory created in the previous step. After downloading, make sure that the permissions on the distribution file allow it to be executed.

```
rtiuser % ls -l
total 84802
-r-xr-x--- 1 testsol sim 32152172 Sep 27 20:20 RTI.1.3NG-V1-
SunOS-5.6-Sparc-SPR0-4.2-dbg-mt.sh*
```

If the file is not executable, change the permissions to allow it to execute:

```
rtiuser % chmod ug+x RTI.1.3NG-V1-SunOS-5.6-Sparc-SPR0-4.2-dbg-mt.sh
```

### 3.3 Installing the Distribution

To install the distribution, run the shell archive:

```
rtiuser % ./RTI.1.3NG-V1-SunOS-5.6-Sparc-SPR0-4.2-opt-mt.sh
```

```
=====
          HLA RUNTIME INFRASTRUCTURE 1.3 NEXT GENERATION
          SOFTWARE INSTALLATION
=====
```

Copyright 1999-2000 Science Applications International Corp., All Rights Reserved.

This computer software has been developed under contract N61339-97-C-0073 in which the regulations from DFARS 252.227-7013 and 252.227-7014 are enforced. This material may be reproduced by or for the U.S. Government pursuant to the copyright license under these DFARS clauses.

```
-----
This script will install the following RTI configuration:
Version : RTI.1.3NG-V1-SunOS-5.6-Sparc-SPR0-4.2-opt-mt
OS      : SunOS-2.6
Compiler : SunProC++-4.2
Build Type : Optimized-Multi-Threaded
-----
x - creating lock directory
Customizing installation for your system ...
Done.
Installation completed successfully!
```

The install program will customize a pair of environment configuration files for your installation. These are used by the sample applications, and are described in more detail below.

You can now proceed to testing the installation to insure that the RTI will work properly on your system. This is described in Section 3.

### 3.3.1 Troubleshooting Installation Problems

If any problems are encountered while installing the distribution, please make sure that the installation is writable. If it is not, the installation may result in errors like this:

```
WARNING: not restoring timestamps. Consider getting and
installing GNU `touch', distributed in GNU File Utilities...
```

```
mkdir: Failed to make directory "_sh28547"; Permission denied
failed to create lock directory
```

Follow the instructions in section 2.3.1 to properly set the permissions on the install directory.

Also note that on Linux platforms, the Bourne shell has been found to lock up when unpacking a large shell archive. As distributed the RedHat Linux RTI distributions use the Korn shell(/bin/ksh).

## 4 Installing the RTI on Windows 98/NT

### 4.1 Create the Installation Folder

The first step is to create an installation folder if desired; if not, the installation wizard will create the directory automatically. There should be sufficient space on the disk where this directory is created to permit installation of the RTI binaries, and for building sample applications. The DMSO Software Distribution Center provides information on disk space requirements for each distribution package.

Windows NT Explorer can be used to create the folder. Click on the **Start** Menu, and then click on **Programs | Windows Explorer** to start NT Explorer. Click on **File | New | Folder** to create the folder.

### 4.2 Download the RTI Distribution

The RTI-NG 1.3 distribution is freely available online from the DMSO Software Distribution Center (SDC). The Windows 98, NT and 2000 distributions use InstallShield to install the software, which will uncompress and install the software in the chosen directory. Selecting the appropriate Windows version of the software from the Web site will download the software to a directory you choose.

### 4.3 Installing the Distribution

Using the Windows Explorer, navigate to the directory chosen during the download phase. Once the downloaded file has been located, double clicking on the file will launch InstallShield. InstallShield will then guide you through the rest of the steps for uncompressing and installing the software. InstallShield will prompt for a directory to install the software into. This directory will be the one created in step one above, or if one was not created in that step, the directory chosen now will be automatically created by InstallShield and the software will be installed in that directory.

## 5 Installing the RTI on a UNIX Platform for VxWorks

### 5.1 Download and Install the RTI Distribution

Download the RTI distribution for VxWorks and follow the instructions for installing the RTI on a UNIX platform outlined in section 3.3.

### 5.2 Prepare a VxWorks Kernel for RTI Operation

#### 5.2.1 Configure Kernel Options

In addition to the standard VxWorks kernel options, the RTI requires that several additional options be set or unset as follows. Some of these are required by the Green Hills C++ compiler.

```
#define INCLUDE_CPLUS
#define INCLUDE_GHS_CPLUS_STD_LOCKS
#define INCLUDE_GHS_CPLUS_EXCEPTIONS
#undef INCLUDE_CPLUS_IOSTREAMS
#define INCLUDE_SHELL
#define INCLUDE_POSIX_ALL
```

These options can be set/unset either by manually editing the configAll.h file or by running Tornado's WindCFG applet. The VxWorks kernel must be rebuilt for the new options to take effect.

#### 5.2.2 Add Green Hills Compiler Support to the Kernel

Refer to “Chapter 3: Installation and Libraries for UNIX Hosts” in the Green Hills Tornado PowerPC Development Guide for instructions on configuring the VxWorks kernel for use with programs produced by the Green Hills compiler. From the matrix depicting the various combinations of language levels and C++ support, choose the entry corresponding to “Exceptions” and “Standard C++”. The makefiles for the RTI demo applications are written with the assumption that all of the Green Hills object files are linked with the kernel. If you opt for a different configuration, you will need to modify the makefiles in order to load the demo applications.

Some copies of the Tornado PowerPC Development Guide contain an important typographical error. The “munch” line in your kernel makefile should look something like the following.

MUNCH = cxxmunch

Note the absence of the “-u” option, which is only for use on Windows platforms.

### 5.2.3 Increase the Task Stack Size

The RTI requires more stack space than the 20k allocated by default for a VxWorks task. A stack size of 30k should suffice for a federate application that does not itself require extensive stack space. The VxWorks *checkStack* function can assist in determining the stack requirements of an application.

The *taskSpawn* function can be used to launch a task with a non-default stack size. Alternatively, the “WDB\_SPAWN\_STACK\_SIZE” definition in *configAll.h* can be modified to permanently increase the amount of stack space allocated by default for tasks spawned by the host-resident shell. The VxWorks kernel must be rebuilt for the new default to take effect.

Users of the Green Hills Multi debugger can increase the default stack size of tasks spawned by the debugger by adding the “-stack <stacksize>” option to the command line used to start *tornserv*.

### 5.2.4 Enable Kernel Support for UDP Multicast (Optional)

The RTI uses UDP multicast for best-effort communication and also to facilitate automatic discovery of the global RTI processes (i.e., *rtiexec* and *fedex*). The default VxWorks network stack does not support UDP multicast, so it is necessary to use the SENS stack if you wish to take advantage of these RTI features. SENS is often installed at the same time as VxWorks/Tornado; consult your system administrator for details. To activate the SENS stack, you must add the following line to the bottom of the appropriate *config.h* file (*not configAll.h*).

```
#define INCLUDE_END
```

This will configure VxWorks to use the new END-style driver instead of the standard BSD-style driver. It is necessary to rebuild the kernel for this option to take effect.

If you are unwilling or unable to use SENS, you must explicitly specify the endpoint of the *rtiexec* process by using the *RtiExecutiveEndpoint* RID option in conjunction with the *-endpoint* command-line option. See the RID file documentation for details. Also, you should ensure that all federation data is sent using the reliable transport.

The amount of buffer space allocated for the network stack is fairly small by default. This can result in an application experiencing ENOBUFS errors if it attempts to sustain a substantial level of UDP multicast throughput. There are three sets of kernel parameters that may need to be adjusted to increase the attainable level of throughput. In order to

use the diagnostic routines mentioned below, you must recompile your kernel with the `INCLUDE_NET_SHOW` option if this option is not already enabled.

In `netBufLib.h`, the definitions `NUM_NET_MBLKS` and `NUM_SYS_MBLKS` specify the number of memory blocks allocated for normal socket data and internal system use, respectively. After experiencing an `ENOBUFS` error, use the functions `mbufShow` and `netStackSysPoolShow` to determine if the error was related to a lack of available normal or system memory blocks, respectively. If the number of times failed to find space reported is greater than zero, increasing the number of the appropriate type of memory block is in order.

Also in `netBufLib.h`, the definitions `NUM_64`, `NUM_128`, `NUM_256`, `NUM_512`, `NUM_1024`, and `NUM_2048` specify the number of the corresponding size of data cluster allocated for normal socket data. If the `mbufShow` function reports a high usage or a low free count for a particular size of data cluster, increasing the number of that size of cluster is in order. Similarly, `netStackSysPoolShow` will indicate whether one or more of the corresponding `NUM_SYS_###` definitions needs to be included.

If the `ifShow` function indicates that output errors are occurring with the network interface, increasing the size of the device output queue is in order. This can be accomplished by running the following function from the command line or from your application.

```
void ifq_maxlen(int len)
{
    struct ifnet *ifp = ifunit("dc0");
    if (ifp == NULL)
        printf("error: ifunit not found\n");
    else
    {
        printf("set ifq maxlen %d -> %d\n", ifp->if_snd.ifq_maxlen, len);
        ifp->if_snd.ifq_maxlen = len;
    }
}
```

The default value of the device output queue is 50.

UDP multicast users should also establish a multicast route by calling the `routeAdd` function with "224.0.0.0" as the destination and the IP address of the target's primary network interface as the gateway. The necessary function calls to your VxWorks kernel initialization routines or to your shell startup script.

### 5.3 Miscellaneous VxWorks Installation/Development Notes

Also, please see the Release Notes for important limitations of the RTI for VxWorks.

### 5.3.1 RTI Stack Objects

Due to the lack of a traditional process space on VxWorks, several objects that are process-wide static objects on other RTI platforms must be instantiated on the stack by the program entry function(s) on VxWorks. This is accomplished by including the header file `ace/OS.h` and adding the statement `ACE_MAIN_OBJECT_MANAGER;` as the first statement of the VxWorks program entry function(s).

### 5.3.2 Slow Application Load Times

The default size of target memory cache used by the Tornado target server is not large enough to allow the relocation of RTI applications to be performed entirely on the host. To avoid very long load times, use the target server's `-m` command-line option to increase the size of the target memory cache. The target memory cache resides on the host, so enlarging it does not consume any memory on the target itself.

### 5.3.3 System Clock Rate and Tick

The VxWorks implementation of *select* deviates from the traditional *select* behavior in that it treats the timeout value as an upper bound on the time to wait rather than a lower bound. The precision of the *select* timeout is limited to the period of the VxWorks system clock, which runs at 60Hz by default. Calling *tick* with a minimum timeout that is approximately equal to (or less than) the system clock period can result in a situation where *select* always times out immediately, which will cause the RTI to enter into a “busy loop” without accomplishing any work. To obviate this sort of timing problem, it is strongly recommended that all RTI users make one of the following adjustments.

- Use the VxWorks *sysClkRateSet* function to set the system clock rate to a rate greater than 100Hz.
- Add a line similar to the following to the *RTI.ProcessSection.Scheduler* section of the RID file. The minimum polling interval should be somewhat greater than the period of the system clock.

```
(MinimumPollingIntervalDuringTicks 0.02)
```

## 6 Testing the RTI on Unix Platforms

This section describes a simple procedure for testing the RTI installation on your network, using the helloWorld sample program. The helloWorld program exercises many of the commonly used HLA services.

Users of the RedHat Linux, Solaris, and IRIX distributions can use the procedure described in this section to test out the RTI in your environment. Caveats for particular platforms are noted where appropriate.

### 6.1 Environment Configuration

The sample applications provided with the RTI require that either the rtienv.csh or the rtienv.sh script is run to configure your shell environment. This script will set the following variables in your shell environment:

- **RTI\_HOME** – set to the location of the RTI installation directory (e.g., /home/data1/rti/RTI1.3NG-V1)
- **RTI\_BUILD\_TYPE** – set to the particular build type for the distribution (e.g., SunOS-5.6-Sparc-SPRO-4.2-opt-mt)
- **LD\_LIBRARY\_PATH** or **LD\_LIBRARYN32\_PATH** – search path for shared object libraries. Setting this variable is required for *any* federate using the RTI-NG 1.3 software. It should include \$RTI\_HOME/\$RTI\_BUILD\_TYPE/lib.

To set up your environment to build and run the sample applications:

```
rtiuser % cd RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-4.2-opt-mt/config
```

C Shell and TCSH Users:

```
rtiuser % source rtienv.csh
```

Bourne Shell and Korn Shell Users:

```
rtiuser % ./rtienv.sh
```

Additionally, your environment should be set up properly for your compiler. This typically includes making sure that your C++ compiler executable (e.g., CC) is on your path, and that the LD\_LIBRARY\_PATH includes the appropriate C++ library directories. Consult your compiler documentation for more details.

## 6.2 Build the helloWorld Application

The helloWorld application is located in the apps directory of the RTI distribution. To build helloWorld:

```
rtiuser % cd $RTI_HOME/$RTI_BUILD_TYPE/apps/helloWorld
```

Invoke the gmake utility (provided with the distribution) to compile and link the executable:

```
rtiuser % ../../bin/gmake

cd src; /home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-4.2-opt-
mt/apps/helloWorld/../../bin/gmake

gmake[1]: Entering directory `/home/testsol/RTI1.3NG-V1/SunOS-5.6-
Sparc-SPRO-4.2-opt-mt/apps/helloWorld/src'

Compiling helloWorld.cpp ...

CC -c -fast -mt -features=castop -D_POSIX_C_SOURCE=199506L -
D__EXTENSIONS__ -D_POSIX_PTHREAD_SEMANTICS -I../include -
I/home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-4.2-opt-mt/include
helloWorld.cpp -o helloWorld.o

Compiling Country.cpp ...

CC -c -fast -mt -features=castop -D_POSIX_C_SOURCE=199506L -
D__EXTENSIONS__ -D_POSIX_PTHREAD_SEMANTICS -I../include -
I/home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-4.2-opt-mt/include
Country.cpp -o Country.o

Compiling HwFederateAmbassador.cpp ...

CC -c -fast -mt -features=castop -D_POSIX_C_SOURCE=199506L -
D__EXTENSIONS__ -D_POSIX_PTHREAD_SEMANTICS -I../include -
I/home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-4.2-opt-mt/include
HwFederateAmbassador.cpp -o HwFederateAmbassador.o

Linking ../data/helloWorld ...

CC helloWorld.o Country.o HwFederateAmbassador.o -o
../data/helloWorld -mt -L/home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-
SPRO-4.2-opt-mt/lib -R/home/testsol/RTI1.3NG-V1/SunOS-5.6-Sparc-SPRO-
4.2-opt-mt/lib -lRTI -lfedtime -L/home/testsol/RTI1.3NG-V1/SunOS-5.6-
Sparc-SPRO-4.2-opt-mt/lib -lorbsvcs -lTAO -lACE -lsunmath_mt -lposix4
-lpthread

gmake[1]: Leaving directory `/home/testsol/RTI1.3NG-
V1/SunOS-5.6-Sparc-SPRO-4.2-opt-mt/apps/helloWorld/src'
```

The helloWorld executable will be created in the 'data' subdirectory, where the RTI FED and RID files are located.

### 6.3 Execute helloWorld

The syntax for the helloWorld application is

```
./helloWorld <Country Name> <Initial Population> [<Number of Ticks>]
```

where

<country name> is the name of the federate,

<initial population> is a starting population for the federate,

<number of ticks> is the number of time advance request/grant cycles for which the federate will execute.

To execute helloWorld, you will need to open three windows on your system. Each should be properly configured to run RTI sample applications, as described above.

In one window, start the rtiexec as follows:

```
rtiuser % $RTI_HOME/$RTI_BUILD_TYPE/bin/rtiexec
```

In another window, start one helloWorld federate:

```
rtiuser % cd data
rtiuser % helloWorld fed1 10 100
```

In another window and start a second helloWorld federate. If you have access to another machine, you optionally can rlogin to that machine to start the second federate. If you choose not to do this, both federates will execute on the local host.

```
rtiuser % cd data
rtiuser % helloWorld fed2 10 100
```

You should see output similar to the following:

```
FED_HW: CREATING FEDERATION EXECUTION
FED_HW: Note: Federation execution already exists.FederationExecutionAlreadyExists: "RtiFederationHandleGetterUn-created.cpp", line 39:
fed2 938492175 75120 START
FED_HW: JOINING FEDERATION EXECUTION: ./helloWorld
FED_HW: JOINED SUCCESSFULLY: ./helloWorld: Federate Handle = 3
FED_HW: ENABLING ASYNCHRONOUS DELIVERY
FED_HW: ENABLING TIME CONSTRAINT
FED_HW: Time granted (timeConstrainedEnabled) to: 0.000000000
FED_HW: Discovered object 65537
```

```

FED_HW: ENABLING TIME REGULATION WITH LOOKAHEAD = 1.000000000
FED_HW: Time granted (timeRegulationEnabled) to: 591.000000000
FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 1
FED_HW: Country[0] Name: fed2 Population: 10.01 Time: 591.000000000
FED_HW: Country[1] Name: (unknown) Population: 100 Time: 0.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 601.000000000
FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 2
FED_HW: Country[0] Name: fed2 Population: 10.02 Time: 601.000000000
FED_HW: Country[1] Name: (unknown) Population: 100 Time: 0.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 611.000000000

...

FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 99
FED_HW: Country[0] Name: fed2 Population: 11.0401 Time:
1571.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 1581.000000000
FED_HW: RESIGN FEDERATION EXECUTION CALLED
FED_HW: SUCCESSFUL RESIGN FEDERATION EXECUTION CALLED
FED_HW: DESTROY FEDERATION EXECUTION CALLED
FED_HW: SUCCESSFUL DESTROY FEDERATION EXECUTION CALLED
FED_HW: HwFederateAmbassador::~HwFederateAmbassador destructor called
in FED
10 938492177 728713 END
FED_HW: Exiting ./helloWorld.

```

If helloWorld executes successfully, the RTI is ready for use. For information about running applications with the RTI, see Section 4.

## 7 Testing the RTI on Windows NT/2000

This section describes the mechanism to test the installation and run the sample helloWorld application for Windows NT and 2000.

### 7.1 Environment Configuration

The sample applications provided with the RTI require the setting of several environment variables. These environment variables are:

- **RTI\_HOME** – set to the RTI installation directory (e.g., c:\rti\RTI1.3NG-V2)
- **RTI\_BUILD\_TYPE** – set to the particular build type for the distribution (e.g., Winnt-4.0-VC6)
- **PATH** – search path for binaries and Windows dlls (dynamically linked libraries). Setting this variable is required for *any* federate using the RTI-NG 1.3 software. It is set to %RTI\_HOME%\%RTI\_BUILD\_TYPE%\bin

Setting environment variables on Windows NT is accomplished in the following manner:

- 1) Move the mouse button over the icon that is usually called “My Computer” on the Windows NT background. Then press the right mouse button. On the menu that pops up select the **Properties** element.
- 2) A new window will pop up after choosing Properties. On this menu please select the tab that is labeled **Environment**. In the section labeled **User Variables for xxx:** (where **xxx** is your user name) select one of the variables. This will cause the variable to show up in the text field called **Variable:** and its value to be displayed in the **Value:** field.
- 3) Click the mouse in the **Variable:** section, delete the text and replace it with the text RTI\_HOME. Then, in the **Value:** field, type in the name of the directory where you installed the RTI from the install steps above (e.g., c:\RTI).
- 4) Finally, click on the Set button.

Repeat the steps above for setting the RTI\_BUILD\_TYPE, except that in the **Value:** field, type the text Winnt-4.0-VC6 and click on set.

In almost all cases, there will already be a **PATH** environment variable defined in the User variables area. In this case, simply click on that Variable, which will display the variable name in the **Variable:** text box and the current value of that variable in the Value: field. Click the mouse in the **Value:** field and move to the end of the text by hit-

ting the end key. Then type a semicolon (;) followed by the path to the RTI concatenated with the value of the RTI\_BUILD\_TYPE and the word bin. So, if the RTI had been installed in c:\RTI, and the PATH variable originally contained the string:

```
c: \Windows\System32
```

then, after adding the required RTI paths, the value of the PATH variable should be:

```
c: \Windows\System32; c: \RTI \Winnt-4.0-VC6\bin
```

## 7.2 Building the helloWorld Application

In order to build the helloWorld included application, which tests that the RTI has been installed correctly and all the environment variables are correctly set, simply use the Windows NT Explorer to navigate to the directory where the RTI was installed. Then continue navigating into the Winnt-4.0-VC6 directory, then into the apps directory and finally into the helloWorld directory. Finally, double click on the file named helloWorldDistrib.dsw. If your compiler is correctly installed, it will then startup and load the helloWorld project. When it has finished loading, simply click on the **Build** menu item and chose the menu item **Rebuild All**. This will build the helloWorld executable and place it in the data directory where the RTI.rid and helloWorld.fed files are located. If the build completes without any errors, it indicates that the RTI files have been installed in the correct locations and that the RTI\_HOME and RTI\_BUILD\_TYPE environment variables have been set correctly.

## 7.3 Execute helloWorld

To execute the helloWorld executable, start three Command Prompts by choosing the **Start | Program Files | Command Prompt** three times. The first step is to start the rtiexec process. To do this, in the first Command Prompt window, change directory to the bin directory located in the Winnt-4.0-VC6 directory. Then type the following command:

```
C: \RTI \Winnt-4.0-VC6\bin>rtiexec
```

Once this window displays the text : rtiexec entering orb event loop, select the second Command Prompt window. In this window, change directory to the data directory located in the helloWorld directory in the RTI distribution. The syntax for the helloWorld application is:

```
./helloWorld <Country Name> <Initial Population> [<Number of Ticks>]
```

where

<country name> is the name of the federate,

<initial population> is a starting population for the federate,

<number of ticks> is the number of time advance request/grant cycles for which the federate will execute.

To execute helloWorld, type the following in the command prompt window:

```
C: \RTI \Winnt-4.0-VC6\apps\helloWorld\data> helloWorld fed1 10 200
```

In the final command prompt, change directory to the data directory in the helloWorld directory in the RTI distribution. Then type the following:

```
C: \RTI \Winnt-4.0-VC6\apps\helloWorld\data> helloWorld fed1 10 200
```

If both helloWorld applications run successfully with similar output to that shown in Section 6.3, then the RTI has been installed correctly and all three environment variables have been set correctly.

## 8 Testing the RTI on Windows 98

This section describes the mechanism to test the installation and run the sample helloWorld application for Windows 98.

### 8.1 Environment Configuration

The sample applications provided with the RTI require the setting of several environment variables. These environment variables are:

- **RTI\_HOME** – set to the RTI installation directory (e.g., c:\rti\RTI1.3NG-V3)
- **RTI\_BUILD\_TYPE** – set to the particular build type for the distribution (e.g., Win98-2-VC6)
- **PATH** – search path for binaries and Windows dlls (dynamically linked libraries). Setting this variable is required for *any* federate using the RTI-NG 1.3 software. It is set to %RTI\_HOME%\%RTI\_BUILD\_TYPE%\bin

Setting environment variables on Windows 98 is accomplished in the following manner:

- 1) Using the Windows Explorer or a MS-DOS window, open the autoexec.bat file for editing.
- 2) Add the following lines to your autoexec.bat file:

```
set RTI_BUILD_TYPE=Win98-2-VC6  
  
set RTI_HOME=c:\progra~1\dms0\rti1~1.3ng\
```

Note: These settings assume that the RTI was installed in the c:\Program Files\DMSO\RTI1.3NGv3\Win98-2-VC6 directory structure. In addition, it is recommended that you use MS-DOS naming criteria in setting your environment variables.

In almost all cases, there will already be a **PATH** environment variable defined in the autoexec.bat file. In any case, add the following line next to your autoexec.bat file:

```
path=%RTI_HOME%\%RTI_BUILD_TYPE%\bin;%path%
```

Note: If you encounter problems with the path statement try replacing the environment variables with the actual path lines (e.g. path=c:\progra~1\dms0\rti1~1.3ng\Win98-2-VC6) .

Save the amended autoexec.bat file and reboot your computer system to activate the new settings.

## 8.2 Building the helloWorld Application

In order to build the helloWorld included application, which tests that the RTI has been installed correctly and all the environment variables are correctly set, simply use the Windows Explorer to navigate to the directory where the RTI was installed. Then continue navigating into the Win98-2-VC6 directory, then into the apps directory and finally into the helloWorld directory. Finally, double click on the file named helloWorldDistrib.dsw. If your compiler is correctly installed, it will then startup and load the helloWorld project. When it has finished loading, simply click on the Build menu item and chose the menu item Rebuild All. This will build the helloWorld executable and place it in the data directory where the RTI.rid and helloWorld.fed files are located. If the build completes without any errors, it indicates that the RTI files have been installed in the correct locations and that the RTI\_HOME and RTI\_BUILD\_TYPE environment variables have been set correctly.

## 8.3 Execute helloWorld

To execute the helloWorld executable, start three Command Prompts by choosing the **Start | Program Files | Command Prompt** three times. The first step is to start the rtiexec process. To do this, in the first Command Prompt window, change directory to the bin directory located in the Win98-2-VC6 directory. Then type the following command:

```
C: \RTI \Win98-2-VC6\bin>rtiexec
```

Once this window displays the text : rtiexec entering orb event loop, select the second Command Prompt window. In this window, change directory to the data directory located in the helloWorld directory in the RTI distribution. The syntax for the helloWorld application is:

```
./helloWorld <Country Name> <Initial Population> [<Number of Ticks>]
```

where

<country name> is the name of the federate,

<initial population> is a starting population for the federate,

<number of ticks> is the number of time advance request/grant cycles for which the federate will execute.

To execute helloWorld, type the following in the command prompt window:

```
C: \RTI \Win98-2-VC6\apps\helloWorld\data> helloWorld fed1 10 200
```

In the final command prompt, change directory to the data directory in the helloWorld directory in the RTI distribution. Then type the following:

```
C: \RTI \Win98-2-VC6\apps\helloWorld\data> helloWorld fed1 10 200
```

If both helloWorld applications run successfully with similar output to that shown in Section 6.3, then the RTI has been installed correctly and all three environment variables have been set correctly.

## 9 Testing the RTI on VxWorks

### 9.1 Environment Configuration

The sample applications provided with the RTI require that either the `rtienv.csh` or the `rtienv.sh` script be run to configure your development shell environment on the host. This script will set the following variables in your shell environment.

- **RTI\_HOME** – set to location of the RTI installation directory (e.g. `/home/RTI-1.3NGv3`)
- **RTI\_BUILD\_TYPE** – set to the particular build type for the distribution (e.g. `VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-mt`)

To set up your environment to build and run the sample applications:

```
rtiuser % cd RTI-1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-mt/config
```

C Shell and TCSH Users:

```
rtiuser % source rtienv.csh
```

Bourne Shell and Korn Shell Users:

```
rtiuser % ./rtienv.sh
```

Additionally, your environment should be set up properly for your compiler. This typically entails making sure that your C++ compiler executable (`cxvxppc`) is in your path, and that your Tornado environment variables (e.g., `WIND_BASE`) are set up correctly.

### 9.2 Building the helloWorld Application

To build hello world, go to the `apps` directory of the installation and invoke the `gmake` executable provided with the distribution.

```
rtiuser % cd $RTI_HOME/$RTI_BUILD_TYPE/apps/helloWorld
/home/RTI-1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-mt/apps/helloWorld % gmake
cd src; /home/RTI-1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-mt/apps/helloWorld/gmake
gmake[1]: Entering directory `/home/RTI-1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-mt/apps/helloWorld/src'
```

```
Compiling HelloWorld.cpp ...
```



```

mtrans: warning: unrecognized symbol type: k:
s.2D5.2E3.2E1.2DPPC.2DGreenHills.2D1.2E8.2E9.2Ddbg.2Dmt.2Fobj.2FRtiOw
nershipManager..0 (ignored)

gmake[1]: Leaving directory `/home/ RTI -1.3NGv3/VxWorks-5.3.1-PPC-
GreenHills-1.8.9-opt-mt/apps/helloWorld/src'

```

The helloWorld executable will be created in the data subdirectory, where the FED and RID files are located.

### 9.3 Execute helloWorld

1. Run the rtiexec process on a Windows or UNIX host.
2. Load the helloWorld image onto the VxWorks target. Using a host-resident shell, the load commands will be something like the following.

```

cd "/home/ RTI -1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-
mt/apps/helloWorld/data"
ld < helloWorld

```
3. Change from the target's working directory to the helloworld's data directory, using a command similar to the following.

```

chdir "hostname:/home/ RTI -1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-
1.8.9-opt-mt/apps/helloWorld/data"

```

Note that the target's working directory is independent of the shell's working directory.

4. Execute hello world's entry-point function.

```

hwmmain "fed1", "10", "1000"

```

To test connectivity, execute a second hello world on a different machine. The output of the VxWorks federate should be similar to the following.

```

FED_HW: CREATING FEDERATION EXECUTION
hla39:/home/ RTI -1.3NGv3/VxWorks-5.3.1-PPC-GreenHills-1.8.9-opt-
mt/apps/helloWorld/data/Directive: No such file or directory.
FED_HW: SUCCESSFUL CREATE FEDERATION EXECUTION
fed1 6 16666 START
FED_HW: JOINING FEDERATION EXECUTION: hw_main
FED_HW: JOINED SUCCESSFULLY: hw_main: Federate Handle = 1
FED_HW: ENABLING ASYNCHRONOUS DELIVERY
FED_HW: ENABLING TIME CONSTRAINT
FED_HW: Time granted (timeConstrainedEnabled) to: 0.000000000
FED_HW: ENABLING TIME REGULATION WITH LOOKAHEAD = 1.000000000
FED_HW: Time granted (timeRegulationEnabled) to: 0.000000000
FED_HW:

```

```

FED_HW: helloWorld Event Loop Iteration #: 1
FED_HW: Country[0] Name: fed1 Population: 10 Time: 0.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 10.000000000
FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 2
FED_HW: Country[0] Name: fed1 Population: 10.01 Time:
10.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 20.000000000

...
FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 998
FED_HW: Country[0] Name: fed1 Population: 27.0879 Time:
8722.000000000
FED_HW: Country[1] Name: fed2 Population: 26.2615 Time:
8722.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 8732.000000000
FED_HW:
FED_HW: helloWorld Event Loop Iteration #: 999
FED_HW: Country[0] Name: fed1 Population: 27.115 Time:
8732.000000000
FED_HW: Country[1] Name: fed2 Population: 26.2615 Time:
8722.000000000
FED_HW: Time granted (timeAdvanceGrant) to: 8742.000000000
FED_HW: RESIGN FEDERATION EXECUTION CALLED
FED_HW: SUCCESSFUL RESIGN FEDERATION EXECUTION CALLED
FED_HW: DESTROY FEDERATION EXECUTION CALLED
FED_HW: SUCCESSFUL DESTROY FEDERATION EXECUTION CALLED
FED_HW: HwFederateAmbassador::~HwFederateAmbassador de-
structor called in FED

```

## 10 Running Applications With The RTI

### 10.1 RTI-NG Components

The RTI-NG 1.3 distribution includes four executable programs. The table below describes each of the RTI components.

Table 3 RTI-NG Components

Component	Description
rtiexec	This executable is responsible for launching the fedex for each new federation and contains the centralized information needed to insure the uniqueness of federation names as well as the mechanism for locating and connecting to already running federations.
fedex	This executable is the centralized component that insures the uniqueness of object names and insures that only one save is in progress at any given time. This executable currently also contains the LbtsMaster component that calculates the LBTS values in a time managed execution.
launch	This executable is used by the rtiexec when the fedex is launched on a machine other than the one the rtiexec is located on. In order for this process to work, the launch executable must be run on the machine the fedex will be created on after the rtiexec has been started.
rtiConsole	This executable is provided for the user to check the status of a federation and its federates without joining the federation. It relies on the rtiexec process for information.

### 10.2 RTI Configuration Files

The RTI requires the presence of two configuration files, the FED (Federation Execution Data) file and the RID (RTI Initialization Data) file.

### 10.2.1 FED File

The FED file contains the listing of object and interaction classes used by your federation and by the RTI. A detailed description of this file and its format can be found in the HLA Interface Specification document. The FED file is used during the federation creation phase. The federation designator parameter in the *createFederationExecution* service specifies the name and location of the FED file. An absolute or relative pathname can be specified. If a relative pathname is provided, it will be assumed to be relative to the directory in which the federate executable resides. When the *createFederationExecution()* method is invoked, the FED file is read, parsed, and stored in an internal database. Once this information has been included, the RTI will not reopen the FED file.

Note that the federation designator only needs to be provided by the federate that successfully invokes the *createFederationExecution* service. Subsequently joining federates do not need to provide the federator designator since the contents of the FED file are provided to each subsequent joining federate.

### 10.2.2 RID File

The RID file contains configuration parameters that are used to control the operation of the RTI software. It can be used to configure or tune the RTI to your particular federation execution. However, all parameters have a default setting that is used in the event that a parameter value is not specified in the RID file or a RID file is not present. In the “out-of-the-box” condition, the *RTI.rid* file provided in the distribution is a “commented out” example file. Each configuration parameter in the RID file is documented to describe the purpose of the parameter, the valid range of values for the parameter, and the default value of the parameter.

The RTI software uses the *RTI\_RID\_FILE* environment variable to define the name and location of the RID file. The file location provided may be absolute or relative using the appropriate convention for the particular operating system. The file name is not required to have a special name or prefix, it only needs to be readable by the application and provide the correct syntax. In the event the *RTI\_RID\_FILE* environment is not set, the RTI software will attempt to open a file named “*RTI.rid*” in the directory from which the application was launched. If the *RTI.rid* file is not found, the RTI software will use the default parameters for operations. Note that the RTI Executive Process (*rtiexec*) does not require a RID file and any configuration information is passed in as a command line argument.

The format used for the RID file has only several rules relative to valid parsing. The first item is that any text to the right of the comment token, (two semi-colons, “;”), is ignored by the parser. The next rule is that the left and right parentheses are used for scoping and must always be used in matching pairs.

Within a pair of parentheses there can be the scope name or a parameter name and value pair. The scope name is used to organize parameters that are conceptually related and ensure uniqueness in case a parameter name is used multiple times within different

scopes. If a parameter name is not unique only the last value will be used for the configuration control. The parameter name is case insensitive and the value is parsed as a character string and subsequently interpreted according to the particular parameter type (e.g., integer, floating point, string).

Each RID parameter is identified by a scope name in which the scoping is broken into three major categories according to the granularity of the internal RTI components. The RTI-NG instantiates components when an RTI process is initially started (the first create or join), when a federation comes into existence within the process (first create or join of a new federation), and when a particular federate joins a federation. These scope names are defined below.

- ProcessSection - process level component parameters
- FederationSection - federation level component parameters
- FederateSection - federate level component parameters

It is possible that a RID file used by a particular application will need to support multiple federations and federates within a single process using different RID parameter values for each federation or federate. This RID structure can support this situation by creating a scope within the federation or federate section with the scope name the same as the name of the federation or name of the federate, respectively.

As an example, assume that an application needs to support two different federations named FederationA and FederationB. The RID parameter for the multicast base address for FederationA needs to be different from FederationB. An example RID is shown below where the BaseAddress used for FederationB is "224.100.0.1" and for any other federations the value is "224.2.0.1".

```
(Federati onSecti on
...
  (BaseAddress 224. 2. 0. 1)
  ...
  (Federati onB
    ...
    (BaseAddress 224. 100. 0. 1)
    ...
  )
)
```

### 10.2.3 RID Consistency

In order for the RTI to run properly certain parameters have to be consistent across the federation. When the first federate joins the federation it passes an IDL structure containing the checksum of these “consistency required” RID parameters to the fedex. The fedex locally stores this IDL structure. When additional federates join the federation they request the checksum value from the fedex and make a comparison with the checksum

value of their own RID file. If the values fail to match the RTI throws an Exception listing the mismatched parameters.

The following is a list of RID parameters that must be consistent federation wide:

```
Networking.MulticastOptions.BaseAddress
DataDistribution.Options.BestEffortChannelType
DataDistribution.Options.StaticGridPartitionedStrategyOptions.
MaxNumberOfDataChannelsToUse
DataDistribution.Options.StaticGridPartitionedStrategyOptions.
NumPartitionsPerDimension
Advisories.RelevanceAdvisoryAttributeInstanceHeartbeatInSeconds
Advisories.RelevanceAdvisoryAttributeInstanceTimeoutInSeconds
Advisories.RelevanceAdvisoryInteractionClassHeartbeatInSeconds
Advisories.RelevanceAdvisoryInteractionClassTimeoutInSeconds
Advisories.RelevanceAdvisoryObjectClassHeartbeatInSeconds
Advisories.RelevanceAdvisoryObjectClassTimeoutInSeconds
DataDistribution.Options.ReliableChannelType
Networking.CompressionOptions.Reliable.CompressionType
DataDistribution.StrategyToUse
DataDistribution.
Options.StaticGridPartitionedStrategyOptions.SpaceOptions
FederationExecutive.DirectoryForSaveAndRestoreFiles
FederationExecutive.FilenameToRedirectStdout
FederationExecutive.FilenameToRedirectStderr
TimeIntervalToCheckForUnresponsiveFederationInSeconds
TimeToWaitBeforeDeclaringFederationDeadInSeconds
RtiExecutive.RtiExecutiveEndpoint
Networking.MulticastOptions.MaxAddress
Preallocation.Size
MOM.MomServiceAvailable
TimeManagementTimeToWaitForLbtsCalculationsBeforeErrorInSeconds
```

### 10.3 RTI Communications Overview

An RTI federation is a distributed system that can contain multiple processes communicating across multiple computer hosts or *nodes*. Currently the RTI-NG 1.3 implementation supports IP (Internet Protocol) based network protocols (specifically TCP, Transmission Control Protocol, and UDP, User Datagram Protocol). The connections used for communications are based on sockets that are specified by a network address and port number pair, also referred to as an endpoint.

The network address is defined either using the hostname (e.g., *www.mydomain.com*) or the *dotted decimal* IP address (e.g., 128.9.128.127). The port number is used to distinguish the different processes using IP communications on a single node. The port number is a 16-bit integer in which ports 0 through 1023 have been assigned and controlled by a standard authority and typically not usable by applications. For the purpose of the RTI any unused port between 1024 and 65535 can be used for communication configuration.

For the purposes of RTI configuration the endpoints of the RTI Executive, Federation Executive, and Federate processes can be defined. Typically the computer platform upon

which these processes are run only has a single network interface, and the network address for the endpoint is specified by the hostname or the IP address of that machine. In the case of a multi-homed machine there may be multiple network interfaces in which the network address of the endpoint may correspond to any one of the network interfaces. Each network interface of a multi-homed machine will have a unique network address.

The endpoint of the RTI executive is defined through a command line option to the `rtiexec` process as shown by the following example,

```
rtiexec -endpoint www.mydomain.com:12345
```

If the endpoint option is not specified for the RTI executive it will use the default network interface and a well known port number defined internally by the RTI software.

The endpoint for the Federation Executive and each Federate process is defined in the RID file. The `FederationExecutiveEndpoint` defines the endpoint used by the Federation Executive process and if not specified in the RID the software will use the default network interface and an available port. Likewise, the `FederateEndpoint` defines the endpoint used by the federate application using the RID file and has similar default behavior.

A bootstrapping process is required to enable the different nodes of the RTI system to communicate. This process begins with the user starting the RTI Executive (`rtiexec`) on a particular machine. When the RTI Executive is started a Naming Service is instantiated internally to assist the process of locating the various distributed RTI components. For example, when a Federate process starts it will need to find the RTI Executive and Federation Executive to carry out certain services. The discovery of the Naming Service by other distributed components can be accomplished by using either a multicast discovery protocol or through the specification of the Naming Service endpoint.

The multicast discovery mechanism is useful in a single LAN environment in which all of the machines participating in the federation are able to exchange UDP multicast traffic. In this mechanism any component looking for the naming service will send out a message to a particular multicast address and port number. If properly configured the Naming Service started by the RTI Executive will be listening to that multicast address and port number and perform the necessary “handshake”. The user can configure the particular multicast address and port number used for the discover protocol through command line arguments to the RTI Executive process, as shown in the following example. Without this information the Naming Service will default to a pre-defined multicast address and port number.

```
rtiexec -multicastDiscoveryEndpoint 224.0.0.1:23456
```

When the network configuration can support UDP multicast communications between LANs (Local Area Networks), the multicast discovery protocol will allow a process within one LAN to find the Naming Service on another LAN provided the time-to-live (TTL) is set properly to allow multicast packets to be routed between the LANs. The TTL is set in the RID file using the parameter `TimeToLive`.

If the Federate process is unable to use multicast communications between a Federate Process and the Naming Service the endpoint of the Naming Service can be specified in the RID file. In this mechanism the Federate Process will attempt to make a TCP connection with the Naming Service. The endpoint of the Naming Service is the same as the endpoint used for the RTI Executive Process. To use this mechanism to discover the Naming Service the RTI Executive must use the command line option "-endpoint" to define the network address and port of the Naming Service. The Federate Processes should use the same endpoint in their RID files for the NamingServiceEndpoint parameter.

## 10.4 The RTI Console Application

The rtiConsole application provides the user with basic diagnostic capabilities for an ongoing RTI simulation:

- List all current federation executions
- Check the status of all current federation executions
- Unregister a federation execution from the rtiexec
- List all federates currently joined to a federation execution
- Check the status of all federates currently joined to a federation execution
- Forcibly resign the federate from the federation execution

The rtiConsole application functions by interfacing with the rtiexec process. In order for rtiConsole to function properly, the rtiexec process must be running somewhere that can be reached by the rtiConsole host. For instructions on running rtiexec, refer to the previous section. Starting with v3.1, the rtiConsole will attempt to establish a new connection with the rtiexec process whenever contact has been lost and a new command is entered at the prompt. This means rtiConsole can be started before the rtiexec or that the rtiexec can be stopped and restarted without stopping rtiConsole, as long as the processes are using the same endpoints. A message is printed to the screen whenever a new connection is established.

### 10.4.1 Usage

Included with the RTI-NG distribution is the rtiConsole application. Follow the instructions for installing the RTI-NG distribution and configuring the environment, if you haven't already. You'll find rtiConsole in the same directory as the other RTI-NG executables (rtiexec, fedex and launch).

To run the rtiConsole application, type at the prompt:

```
rtiConsole [arguments]
```

where the optional command-line arguments can be any of the following:

`-rtiexecEndpoint hostName:portNumber`

Specify the endpoint of the `rtiexec` process. [For a definition of the endpoint, consult section 9.2 of this guide.]

`-multicastDiscoveryEndpoint multicastAddress:portNumber`

Specify the multicast discovery endpoint of the `rtiexec` process. [For a definition of the multicast discovery endpoint, consult section 9.2 of this guide.]

`-exec "<console command>"`

Perform the given console command, then exit. This is an alternative to entering the console command at the `rtiConsole` prompt. The accepted console commands are listed in the next section.

`-help`

`-h`

List the accepted command-line arguments and exit.

## 10.4.2 Commands

The user interface for the `rtiConsole` application is a simple console that accepts commands. The prompt for the console looks like this:

```
rti console >
```

At this prompt, you can type in any of the following commands:

```
exit
```

```
quit
```

Exit from the `rtiConsole` application.

```
federation <federation name or handle>
```

Enter the federation context, in order to perform a federation command [listed below]. The command applies to the federation execution that is specified by the second argument, which can be either the name or the handle of the federation execution.

Once you've entered the federation context, the prompt will change to this:

```
federation [<federation name>] >
```

where `<federation name>` is the name of the specified federation. To leave the federation context, use the 'exit' command.

federation <federation name or handle> <federation command>

An alternative way to perform a federation command [listed below]. This method does not enter the federation context; rather, it stays in the rtiConsole context.

help

Print out a list of accepted rtiConsole commands.

help\_tree

Print out a list of accepted rtiConsole and federation commands.

list

List the names and handles of all current federation executions known by the rtiexec process.

status

List the names, handles and status of all current federation executions known by the rtiexec process sorted by process status. The status is one of:

OK: The federation executive is known to the rtiexec process, initialized and responding.

not found: The federation execution is not known to the rtiexec process. This status can occur when in the federation context and the federation is destroyed by the last resigning federate.

not fully initialized: The federation execution is known to the rtiexec process, but is not yet fully initialized and therefore cannot be contacted. This status can occur when the federation execution first starts.

not known: The federation execution status could not be determined. This status can occur if the rtiexec process cannot be contacted.

not responding: The federation execution is known to the rtiexec process and initialized but cannot be contacted. This status can occur when the process crashes.

remove <federation name or handle>

Remove a federation execution from the rtiexec. This removes the federation from the registered list in the rtiexec process and terminates the federation executive process, after asking for verification. Note that this does not provide an orderly shutdown and does not terminate any federate processes.

### 10.4.2.1 Federation commands

The following federation commands can be used only after entering the federation context:

`exit`

`quit`

Exit from the federation context. This will put you back in the `rtiConsole` context, where you can execute the commands listed above.

`help`

List the federation commands.

The following federation commands can be used in one of two ways:

1. After entering the federation context
2. As the additional arguments to the federation `<federation name or handle>` command

`list`

List all federate types and handles for this federation execution.

`status`

List all federate types, handles and status for this federation execution sorted by process status. The status is one of:

**OK:** The federate is known to the federation, initialized and responding.

**not found:** The federate is not joined to the federation.

**not fully initialized:** The federate is known to the federation, but is not yet fully initialized and therefore cannot be contacted. This status can occur when the federate first starts.

**not known:** The federate status could not be determined. This status can occur if the federation execution or `rtiexec` process cannot be contacted.

**not responding:** The federate is known to the federation and initialized but cannot be contacted. This status can occur when the federate exits without resigning or crashes.

remove <federate type or handle>

Forcibly resign a federate in this federation execution. The federate can be specified by either the type or the handle. If there are multiple federates with the same type, rtiConsole will ask you to use the handle instead. This action will not terminate the federate process.

[Note: the federate 'type' is the first argument that the federate passed to joinFederationExecution.]

### 10.4.3 Notes

All Unix versions of the rtiConsole application make use of the GNU readline library, version 4.0. The source code for this library can be obtained from the GNU Project web site at [www.gnu.org](http://www.gnu.org). [Note: the Windows NT version does not use the readline library; so the capabilities listed below do not apply.]

The readline library adds the following capabilities to rtiConsole:

- Command history

To recall previous commands that you have entered into rtiConsole, press the up-arrow key.

- Line editing

The following key combinations will have these effects on a line of text at the rtiConsole prompt:

Ctrl-a:        move cursor to beginning of line

Ctrl-e:        move cursor to end of line

Ctrl-d:        delete character at cursor

- Tab completion of commands

After typing the first few characters of a command, hitting the TAB key will complete the command. If multiple commands match the characters that you've typed so far, a list of these commands will be displayed.

If you've typed a command keyword but have not entered any of its arguments, hitting TAB will explain what the argument is.

## 10.5 RTI Configuration Tool

The Configuration Tool application provides the user with basic data collection capability for diagnosing network configuration problems. The Configuration Tool is controlled by the sys\_info.bat file. This tool will scan the computer system for the following network

configuration/settings and save the results to a file named “SYS\_INFO.TXT” in the current directory:

- Environment Parameters:
- User Domain:
- Logon Server:
- Computer Name:
- User Name:
- OS:
- Processor:
- NUMBER\_OF\_PROCESSORS:
- Disk Status:
- Hostname:
- IP Configuration:
- Net Configuration -- Workstation:
- Net Configuration -- Server:
- Net Status:

### 10.5.1 Usage

Included with the RTI-NG distribution is the Configuration Tool application. Follow the instructions for installing the RTI-NG distribution and configuring the environment, if you haven't already. You'll find `sys_info.bat` in the same directory as the other RTI-NG executables (`rtiexec`, `fedex` and `launch`).

To run the Configuration Tool application, type at the prompt:

1. Change directory to make the RTI executable directory you current directory. (The `$RTI_HOME/$RTI_BUILD_TYPE/bin` directory.)
2. Enter “`sys_info.bat`” (on Windows) or “`sys_info.sh`” (on Unix).

The program will issue a short message, similar to the following, describing its activities:

Probing for system information...  
The name specified is not recognized as an  
internal or external command, operable program or batch file.

The system information has been written to a file named SYS\_INFO.TXT.  
Please review this information, and note any discrepancies between  
what was found and what you think should have been found.

You may be asked to email the contents of SYS\_INFO.TXT to RTI Support  
to assist in configuring or diagnosing your RTI installation.

When complete, it will have created a file named “SYS\_INFO.TXT” in the current directory. On Windows, this file will automatically display in your default text editor (typically Notepad), for your review.

The file, “SYS\_INFO.TXT”, contains information that may aid in configuring your RTI setup. The information may also help RTI Support personnel in the event that you require assistance diagnosing problems encountered during RTI installation.

## 10.5.2 Notes

The main program, sys\_info.bat/sys\_info.sh, is primarily a data collection tool. Knowledgeable network technicians can use this collected data to determine network configuration and aid in network configuration problem resolution.

Another program provided in the \$RTI\_HOME\RTI\_BUILD\_TYPE\etc directory resolut.exe, does some intuitive checking to see if a given hostname resolves properly. If you run it as (e.g) “resolut.exe hla54“, it will display whether the provided hostname is “good”, and if so, the corresponding IP address. In addition, if you execute it as “resolut.exe hla54 206.239.39.2” it will determine whether that hostname and address are both “good” and actually map to each other.

## Appendix A. Platform Specific Information

### 10.6 Solaris 2.6

For Solaris 2.6 with the SPRO 4.2 compiler, the following additional options should be used when compiling and linking with the RTI 1.3NG:

<b>CFLAGS</b>	<b>+= -mt -features=castop</b>
<b>LDFLAGS</b>	<b>+= -mt</b>
<b>LDLIBS</b>	<b>+= -lsunmath_mt -lposix4</b>

For Solaris 2.6 with the egcs 1.1.2 compiler, only the following addition is necessary to the example makefile:

<b>CFLAGS</b>	<b>+= -D_REENTRANT</b>
---------------	------------------------

### 10.7 Required System Patches for Solaris 2.6 Platforms

Latest patch list:

105181-10	105210-17
105216-03	105284-18
105356-07	105357-02
105375-09	105379-05
105393-07	105395-03
105401-16	105403-01
105407-01	105464-01
105490-05	105552-02
105558-03	105562-03
105566-05	105568-11
105580-08	105591-02
105600-06	105615-04
105621-09	105633-12
105637-01	105642-05

105665-03	105667-02
105669-04	105703-07
105720-06	105741-05
105755-06	105778-01
105780-01	105786-06
105792-02	105797-05
105798-03	105800-05
105837-02	105926-01
106040-10	106049-01
106125-05	106141-01
106193-03	106222-01
106226-01	106235-02
106242-01	106257-04
106271-04	106301-01
106439-02	106448-01
106522-01	106828-01
106929-01	

The list of Solaris patches on your system can be obtained by entering

```
% showrev -p
```

For the SunOS-5.6-Sparc-egcs-1.1.2-opt-mt configuration, the above patches are still required. Additionally, the GNU binutils-2.9.1 (available from <ftp://ftp.gnu.org/gnu/binutils>) and egcs-1.1.2 (available from <http://egcs.cygnum.com>) are required.

## 10.8 Solaris 2.7

For Solaris 2.7 with the SPRO 5.0 and gcc-2.95.2 compiler, the following additional options should be used when compiling and linking with the RTI 1.3NG:

<b>CFLAGS</b>	<b>+= -mt -features=castop</b>
<b>LDFLAGS</b>	<b>+= -mt</b>
<b>LDLIBS</b>	<b>+= -lsunmath_mt -lposix4</b>

## 10.9 Required System Patches for Solaris 2.7 Platforms

Latest patch list:

106541-08	106793-03
107544-03	107451-02
107454-03	107792-02
108301-01	108482-01
106944-02	107456-01
107893-05	108374-01
106934-03	108219-01
107587-01	106978-09
108662-01	107887-08
107022-05	108221-01
106725-02	106952-01
107038-01	107885-06
108343-02	106327-06
107337-01	106960-01
107115-03	107636-03
107359-02	107684-01
106748-04	107972-01
107171-04	107259-01
107311-09	107311-10
107295-01	107357-07
107357-09	107289-05
107355-04	

The list of Solaris patches on your system can be obtained by entering

```
% showrev -p
```

## 10.10 IRIX 6.5

The IRIX 6.5 RTI is built using intermediate release 3 (6.5.3m) of IRIX 6.5. It is uncertain whether earlier releases of IRIX 6.5 would need to be upgraded. The release level can be determined using the command

```
% uname -R
```

No special compile or link options are necessary.

## 10.11 Linux 2.2

For Linux 2.2, the following additional options should be used when compiling and linking with the RTI 1.3NG:

```
CFLAGS += -D_REENTRANT
```

bash-1.14 (which is shipped as /bin/sh with Redhat Linux) may hang when asked to unpack very large files (such as libRTL.so) from a shar archive. If you run into this problem please use ksh which also ships with Redhat Linux to unpack the shell archive successfully.

## 10.12 Required System Patches for Linux 2.2 Platforms

Redhat-5.2 with all recommended updates from <ftp://ftp.redhat.com> . egcs-1.1.2 (available from

<http://egcs.cygnum.com>).

## 10.13 Windows NT 4.0

To run the Windows NT version of the RTI 1.3NG distribution, you must install Windows NT Service Pack 3. Type "winver" at an MS-DOS prompt to determine if Service Pack 3 has been installed on the machine.

Service Pack 3 of the Microsoft Visual Studio compiler is also required to run the RTI 1.3NG distribution.

These files can be downloaded from the Microsoft web site (<http://www.microsoft.com>).