RTI Connext DDS

Core Libraries

Custom Support for
INTEGRITY 5.0.11 Platforms
on MPC 8349 CPUs

Version 5.3.0

Technical Support
Real-Time Innovations, Inc.
232 E. Java Drive
Sunnyvale, CA 94089
Phone: (408) 990-7444
Email: support@rti.com
Website: https://support.rti.com/

Contents

Custom Support for INTEGRITY 5.0.11 Platforms

Chapter 1 Supported Platforms

This document supplements the RTI Connext DDS Core Libraries Release Notes and RTI Connext DDS Core Libraries Platform Notes. It provides information specifically for the platform in Table 1 Custom Supported INTEGRITY 5.0.11 Platforms .

Table 1 Custom Supported INTEGRITY 5.0.11 Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
INTEGRITY 5.0.11	MPC8349	GHnet2 TCP/IP stack	mpc8349Inty5.0.11.mds8349

Chapter 2 Transports

Supported and enabled by default:

- Shared memory
- UDPv4

Not supported:

- UDPv6
- TCP/IPv4

2.1 Using Shared Memory on INTEGRITY 5.0

Core Libraries uses the single address-space POSIX library to implement the shared-memory transport on the INTEGRITY 5.0 operating system.

To use shared-memory, you must configure your system to include the POSIX shared-memory library. The **posix_shm_manager** must be running in an "AddressSpace" solely dedicated to it. After building any Core Libraries application that uses shared memory, you must use the **intex** utility (provided with the INTEGRITY development environment) to pack the application with multiple address-spaces: one (or more) to contain the Core Libraries application(s), and another one to contain the **posix_shm_manager**.

Core Libraries applications will run on a target without the **posix_shm_manager**, but the POSIX functions will fail and return **ENOSYS**, and the participants will fail to communicate through shared memory.

To include the POSIX Shared-Memory Manager in its own Address Space:

The project files generated by *rtiddsgen* for MULTI will create the shared-memory manager for you. Please follow these steps:

1. Specify the path to your INTEGRITY distribution in the **_default.gpj** top-level project file by adding the following line (modify this according to the path to your INTEGRITY distribution):

   ```
           -os_dir=/local/applications/integrity/integrity-5.0.4
   ```

2. Build the project.
3. Before running your Core Libraries application on a target, download the **posix_shm_manager** file (generated by the build) onto the target.

The POSIX Shared Memory Manager will start automatically after the download and your applications will be able to use shared memory.

Notes:

- Only *one* posix_shm_manager is needed on a particular target. INTEGRITY offers the option of building this posix_shm_manager *inside* the kernel. Please refer to the INTEGRITY documentation.
- If you are already using shared memory through the POSIX library, there may be a possible conflict.
- INTEGRITY 5 has two different types of POSIX library: a single-address space one (or 'light') and another one (complete POSIX implementation). Core Libraries uses the first one, but will work if you are using the complete POSIX implementation.

2.1.1 Smaller Shared-Memory Receive-Resource Queue Size

INTEGRITY's shared-memory pluggable transport uses the shared-memory POSIX API. This API is part of the standard INTEGRITY distribution and is shipped as a library. The current version (5.0.4) of this library uses a hard-coded value for the total amount of memory that can be shared with an address space. This limits the overall buffer space that can be used by the *DomainParticipants* within the same address space to communicate over shared memory with other *DomainParticipants*.

To allow more *DomainParticipants* to run within the same address space, we reduced the default size of the queue for each receive resource of the shared memory transport. The queue size is reduced to 8 messages (the default for other platforms is 32). This change only applies to INTEGRITY architectures and this default value can be overwritten through the shared memory transport QoS.

2.1.2 Shared Memory Limitations on INTEGRITY 5.0

If several applications are running on the same INTEGRITY node and are using shared memory, once an application is stopped, it cannot be restarted. When the application is stopped (gracefully or ungracefully), any new application on the same domain index within the same domain will fail to start until the shared memory manager is also restarted.

Additionally, if the application is stopped ungracefully, the remaining applications will print several error messages such as the following until Core Libraries purges the stopped application from its database:

```
Resource Manager send error = 0x9
```

This error message is logged from INTEGRITY's POSIX shared memory manager, *not* from Core Libraries. The error message is benign and will not prevent the remaining applications from communicating with each other or with application on other nodes.

The workaround is to either restart the stopped application with a different participant index or shut down all the other applications and the shared memory manager, then restart everything.

Chapter 3 Features

These features are supported:

- Multicast

These features are not supported:

- Modern C++ API
- Request-Reply communication pattern
- Monotonic clock
- Control of CPU core affinity for RTI threads
- RTI Distributed Logger

3.1 Single NIC Support

As Green Hills explains, the evaluation BSP for INTEGRITY 5.0.11 only enables a single NIC. Therefore Connext DDS only supports a single NIC on the mpc8349Inty5.0.11.mds8349 platform.

Chapter 4 Compiling and Running

This section provides specific instructions on how to compile, link, and run Core Libraries applic-ations on the mpc8349Inty5.0.11.mds8349 target platform.

Table 1 Building Instructions lists the compiler flags and the libraries you will need to link into your application.

Table 2 Running Instructions provides details on the environment variables required to be set at run time.

Table 3 Library-Creation Details provides details on how the libraries were built. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debug-ging.

Table 1 Building Instructions

API	Library Format	Required RTI Libraries[a] [b] [c]	Required System Libraries[d]	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	libsocket.a libnet.a libposix.a	RTI_INTY
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a (libnddscppzd.dba libnddsczd.dba libnddscorezd.dba) librticonnextmsgcppzd.dba		
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a		
	Static Debug	libnddsczd.a libnddscorezd.a (libnddsczd.dba libnddscorezd.dba) librticonnextmsgczd.a		

Table 2 Running Instructions

RTI Architecture	Required Environment Variables
All INTEGRITY 5 architectures	None

[a] The C/C++ libraries are in <NDDSHOME>/lib/<architecture> (where <NDDSHOME> is where Connext DDS is installed, such as /home/your user name/rti_connext_dds-5.x.y).

[b] The *rticonnextmsg* library only applies if you have the Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the Connext DDS Core package type.

[c] The *.dba files contain the debugging information. You can link without these, as long as they are in the same directory as the matching *d.a file (so that the MULTI® IDE can find the debug information).

[d] Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For details, see the API Reference HTML documentation or contact support@rti.com.

Table 3 Library-Creation Details

RTI Architecture	Library Format	Compiler Flags Used by RTI
mpc8349Inty5.0.11.mds8349	Static Release	-bspname=mds8349 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=-DTARGET="mpc8349Inty5.0.11.mds8349\" -c
	Static Debug	

4.1 Diagnostics on INTEGRITY Systems

Connext DDS libraries for INTEGRITY platforms use **consolestring()**, which prints debugging information to the serial console when available. Using the serial console as opposed to the target I/O window (host I/O) is generally recommended. Host I/O will affect the real-time performance of the target.

For more information on **consolestring()**, please refer to the *INTEGRITY Development Guide.*

4.2 Socket-Enabled and POSIX-Enabled Threads are Required

Connext DDS on INTEGRITY platforms internally relies on the POSIX API for many of its system calls. As a result, any thread calling Connext DDS must be POSIX-enabled. By default, the 'Initial' thread of an address space is POSIX-enabled, provided the address space has been linked with libposix.a. Additional user threads that call Connext DDS must be spawned from the Initial thread using pthread_create. Only then is the created thread also POSIX-enabled. Note that tasks created at build time using the Integrate utility are *not* POSIX-enabled.

Furthermore, threads calling Connext DDS must be socket-enabled. This can be achieved by calling InitLibSocket() before making any Connext DDS calls and calling ShutdownLibSocket before the thread terminates. Note that an Initial thread is, by default, socket-enabled when the address space is linked with libsocket.a. Please refer to the *INTEGRITY Development Guide* for more information.

4.3 Using rtiddsping and rtiddsspy on PowerPC INTEGRITY Systems

rtiddsping and *rtiddsspy* are provided as executables, and therefore are BSP-dependent. You will not be able to run them successfully on your target if it is not compatible with the BSP listed in the architecture name (mpc8349Inty5.0.11.mds8349). Please refer to your hardware documentation for peripheral compatibility across BSPs.

Chapter 5 Known Issues

5.1 Delay when Writing to Unreachable Peers

On INTEGRITY systems, if a publishing application's initial peers list includes a nonexistent (or simply unreachable) host, calls to **write()** may block for approximately 1 second.

This long block is caused by the stack trying to resolve the invalid/unreachable host. Most IP stacks do not block the sending thread because of this reason, and you may include invalid/unreachable hosts in your initial-peers list. If you find that your stack does block the sending thread, please consult your IP stack vendor on how to change its behavior.

[RTI Issue ID CORE-1637, Bug # 10768]

5.2 Compiler Warnings Regarding Unrecognized #pragma Directives

Building Core Libraries projects for INTEGRITY causes the compiler to produce several warnings about #pragma directives not recognized in some Core Libraries header files. For example:

```
Building default.bld
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 926:
warning: unrecognized #pragma
  #pragma warning(push)
          ^
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 927:
warning: unrecognized #pragma
  #pragma warning(disable:4190)
          ^
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 945:
warning: unrecognized #pragma
  #pragma warning(pop)
          ^
```

These warnings do not compromise the final application produced and can be safely ignored.

5.3 Warning when Loading Connext DDS Applications on INTEGRITY Systems

When a Core Libraries application compiled with the *rtiddsgen*-generated project files is loaded on an INTEGRITY 5.0.x target, the following warning appears:

```
"Warning: Program is linked with libc.so POSIX signals and cancellation will not work."
```

The Core Libraries libraries do not use the additional features provided by the full POSIX implementation, therefore the warning can safely be ignored.

This warning is due to the fact that the *rtiddsgen*-generated project files use the Single AddressSpace POSIX library by default, not the full POSIX implementation on INTEGRITY (POSIX System). The Core Libraries libraries only require Single AddressSpace POSIX to function correctly, but will still work if you are using the POSIX System.

The message indicates that items such as inter-process signaling or process-shared semaphores will not be available (more information can be found in the *INTEGRITY Libraries and Utilities User's Guide*, chapter "Introduction to POSIX on INTEGRITY").

5.4 Linking with ‘libivfs.a’ without a File System

If you link your application with **libivfs.a** and are using a system that does not have a file system, you may notice the application blocks for 2 seconds at start up.