RTI Recording Service

Release Notes

Version 5.1.0

Your systems. Working as one.

Trademarks

Copy and Use Restrictions

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

Release Notes

1 System Requirements

RTI® *Recording Service* is supported on these platforms:

❑ Linux® Platforms:

- CentOS 5.4, 5.5, 6.0, 6.2 - 6.4
- Fedora® 12
- Red Hat® Enterprise Linux 5.0-5.2, 5.4, 5.5, 6.0 - 6.4
- SUSE® Linux Enterprise Server 11 SP2 (2.6 and 3.x kernel)
- Ubuntu® Server 12.04 LTS

❑ All Windows® platforms listed in the *RTI Core Libraries and Utilities Release Notes* with the same version number. All 64-bit platforms run in 32-bit mode.

For more information on these platforms, see the *RTI Core Libraries and Utilities Release Notes* and *Platform Notes*.

2 Additional Libraries Needed on Select 64-Bit Platforms

If you are installing *Recording Service* on an Ubuntu 64-bit platform or a Red Hat Enterprise Linux 5 or 6 64-bit platform, you will need to install a set of 32-bit libraries *before* installing *Recording Service*:

❑ On Red Hat Enterprise Linux 5 and 6 64-bit platforms, run this command (all on one line); you will need *root* permission:

```
yum install glibc.i686 libX11.i686 gtk2.i686 libXtst.i686
   ncurses-libs.i686 PackageKit-gtk-module.i686 libcanberra-gtk2.i686
```

❑ On Ubuntu 64-bit platforms, run this command (you will need *root* permission):

```
apt-get install libc6 ia32-libs-multiarch
```

3 Compatibility with Other RTI Products

The *Record* tool supports the standard *Connext* transports (UDPv4, UDPv6, and shared memory), as well as the *RTI Secure WAN Transport* plugins.

Recording Service 5.1.0 is compatible with *RTI Connext* 5.0.0 and higher, as well as *RTI Data Distribution Service* 4.5[b-f], 4.4d, 4.4b, 4.3e and 4.2e[1] except as noted below.

- ❏ *Recording Service* 5.1.0 is not backwards compatible with databases recorded with previous releases of *RTI Recording Service*. This applies to all the tools in *Recording Service*: *Record*, *Replay*, *Converter*, and *Recording Console*.
- ❏ *Recording Service* is not compatible with applications built with *RTI Data Distribution Service* 4.5e and earlier releases when communicating over shared memory. For more information, please see the Transport Compatibility section in the *RTI Core Libraries and Utilities Release Notes*.
- ❏ In *Connext* 5.1.0, the default **message_size_max** for the UDPv4, UDPv6, TCP, Secure WAN, and shared-memory transports changed to provide better out-of-the-box performance. *Recording Service* 5.1.0 also uses the new default value for **message_size_max**. Consequently, *Recording Service* 5.1.0 is not out-of-the-box compatible with applications running older versions of *Connext* or *RTI Data Distribution Service*. Please see the *RTI Core Libraries and Utiltities Release Notes* for instructions on how to resolve this compatibility issue with older *Connext* and *RTI Data Distribution Service* applications.
- ❏ Some changes have been made to the Record and Replay IDL files. See Section 4.4.9.
- ❏ The types of the remote administration and monitoring topics in 5.1.0 are not compatible with 5.0.0. Therefore:
  - The 5.0.0 *Record* and *Replay* shells, *Admin Console* 5.0.0 and *Connext* 5.0.0 user applications performing monitoring/administration are not compatible with *Recording Service* 5.1.0.
  - The 5.1.0 *Record* and *Replay* shells, *Admin Console* 5.1.0 and *Connext* 5.1.0 user applications performing monitoring/administration are not compatible with *Recording Service* 5.0.0.

3.1 Command-Line Options Compatibility

Starting with Release 5.1.0, the *Replay* tool's command-line parameter, **-forceXmlTypes**, is deprecated. The XML type configuration will always be used if it is available.

For details on how the *Replay* tool selects a type definition for a Topic see Section 7.10 in the *Recording Service User's Manual*.

4 What's New in 5.1.0

4.1 New Platforms

This release adds support for the following platforms:

- ❏ CentOS and Red Hat Enterprise Linux 6.2 - 6.4
- ❏ SUSE Linux Enterprise Server 11 SP2 (x86, 3.x kernel)
- ❏ Ubuntu Server 12.04 LTS
- ❏ Windows 8 (32-bit and 64-bit Editions)
- ❏ Windows Server 2012 R2 (64-bit Edition)

1. To support compatibility with 4.2e, please see the *RTI Core Libraries and Utilities Release Notes*.

4.2 Extensible Type Support

4.2.1 Support for Mutable Types

This release includes support for mutable types, as defined in the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification from the Object Management Group (OMG).

Starting with this release, *Recording Service* can record and replay topics where the underlying types are mutable.

4.2.2 Support for Optional Types

This release adds support for optional members, as defined in the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification from the Object Management Group (OMG). Starting with this release, the *Record* and *Replay* tools can work with topics whose underlying types have optional members.

4.2.3 Extensible Type Directives Added to Recorder and Replay IDL Files

The IDL definitions for the remote administration of *Record* and *Replay* tools have been updated to make use of the new Extensible Types capabilities.

4.3 Improvements to Recording Console

4.3.1 New Behavior when Using Console's Pause and Restart Buttons

The behavior of some of the buttons on the *Console* have changed:

❑ When using the *Console* to replay data at a non-default speed, pressing Pause and then Play will resume replay with no change to the replay rate. In the previous release, replay would have changed to the default rate.

❑ If you press the Fast-Forward button after pausing replay (this action replays at a higher rate while the button is pressed) and then release the button, replay will go back to a paused state. (In the previous release, replay would have continued at a normal rate.)

❑ If you pause recording and then press Record, recording will resume. (In the previous release, clicking on the record button in this state would prompt the user to acknowledge restarting the recording.)

4.3.2 New Behavior for Record Button when Recording is Paused

In previous releases, if you paused recording and then clicked on the Record button, the *Console* would prompt and ask if it should restart the recording and record over the file.

In this release, the *Console*'s interface has been slightly changed when paused. Now if you click on the Record button while paused, it will resume recording, just as if you clicked on the Pause/ Resume button.

4.3.3 Changes to Console's Dialog to Create Recording File

The dialog that is used to create a recording file has been changed slightly. The position of the two buttons on the bottom to either **Discard** or **Create File** have been reversed, so that the most natural choice (Create File) appears on the right.

4.3.4 Ability to Drag and Drop Files

To select a configuration file or a recording file, you can continue to use the old method of browsing to a folder and selecting a file, or you can simply drag-and-drop the desired file onto the *Console*'s panels.

4.3.5 Improved Configuration Mode Selector

The icon for selecting a configuration mode in *Recording Console* has been changed to a drop-down menu from which you can choose whether or not to use an external configuration file.

4.4 Other Improvements and Features

4.4.1 Ability to Select What SampleInfo, Metadata and Discovery Fields to Store in Recorded Databases

The *Record* tool now includes the ability to filter in or out the Sample Info, discovery or metadata fields to be recorded. Fields can be specified by using the name, or by using regular expressions to group them. This allows you to save space in the recorded databases, which improves resource utilization.

In addition, the default set of recorded fields has been modified. In previous versions of the tool, all Sample Info, discovery and metadata fields were stored in the database along with the user-data samples. Starting with this version, only the fields required for the recorded databases to be compatible with the *Replay* and *Converter* tools are recorded if no specific field filters are specified. See the *Recording Service User's Manual* for details.

There is also a new boolean flag in the XML configuration, **<replay_compatibility>**, that can be used. When set to true, this new flag ensures that the necessary fields for *Replay* and *Converter* to work are always preserved, regardless of what the field filters are.

4.4.2 Ability to Configure Replay to Use a Non-Default Path Separator

SQLite tables created by *Recording Service* have table names constructed from the topic_name, record_group_name, and domain_name, separated by a "path_separator" character. The *Record* and *Replay* tools use the '**$**' character as the default path_separator character, but while the *Record* tool could be configured to use a different character, the *Replay* tool could not. This deficiency has been corrected. Now you may specify the path_separator character for each replay_database entity, using the XML <path_separator> tag.

The path separator may be any character except '**!**', '~', '\', and '/'. (These are the same characters restricted by the *Record* tool). However, when using non-default path_separators, you must ensure that the path separator character is unique—i.e., not used in any topic_name, record_group_name, or domain_name. Failure to ensure the uniqueness of the path_separator character will most likely cause the *Replay* tool to fail.

4.4.3 Replay and Recording Console Now Autodetect Path Separator from Recorded Database

When a database is loaded by the *Replay* tool or *Recording Console*, the path separator is automatically read from the metatable data.

4.4.4 Ability to Change Record Tool's Partition QoS

For each Record Group in the XML configuration, *Recording Service* creates a DDS Subscriber. There is one Subscriber in each of the Record Group's domains. This release adds a way to change the Partition QoS setting for these Subscribers.

For instance, suppose the *Record* tool needs to record data from producers that are organized into different partitions based on their geographical distribution (different locations are represented by different Partition strings). You can specify a Partition QoS policy via XML configuration when the *Record* tool starts up. But suppose the data producers change location—and thus change their Partition QoS? To keep recording the data, the *Record* tool needs to change its Partition QoS too.

Starting with this release, you can change the Partition QoS for any or all Record Groups by using the new RTI_REMOTECTX_MSG_RECORDER_PARTITION message type. See the updated *RTI Recording Service User's Manual* (Section 5.3.1) for more information.

4.4.5 Ability to Specify SQLite PRAGMA Statements to Execute Before Table Creation

Recording Service now includes the ability to specify SQLite pragma statements to be executed right after the database file is created and before any table or data is inserted. Pragmas are specified using the new **<sqlite_pragmas>** XML setting included in the **<recorder_database>** settings.

This new feature can be used, for example, to change the page size on Windows systems for better efficiency, or to change the journal-mode settings to use Write Ahead Logging (WAL) mode for improved concurrent access to the database while recording.

4.4.6 Improved Exporting of Sequences and Arrays of Octets in Converter

The way sequences and arrays of octets and characters are exported has been improved in *Converter*. Bytes are now shown as their uppercase hex representation and different bytes are separated by dashes, e.g. "FA-05-5B"...

4.4.7 Record and Replay now Include Support for QoS Topic Filters

This release supports setting topic filters in the QoS settings for the *Record* and *Replay* tools.

When defining DataReader QoS settings in the *Record* tool's configuration, you can specify a topic filter in the Topic Group settings. For example:

```
<topic_group name="TopicGroup">
    ...
    <datareader_qos topic_filter="Topic1">
        ...
    </datareader_qos>
</topic_group>
```

The same applies to *Replay's* DataWriter QoS settings within the <output> settings of the <replay_topic> section. For example:

```
<replay_topic>
    ...
    <output>
        <datawriter_qos topic_filter="Topic1">
            ...
        </datawriter_qos>
    </output>
<replay_topic>
```

For information on topic filters, see the *RTI Core Libraries and Utilities User's Manual*.

If you do not specify DataReader or DataWriter settings in the *Record* or *Replay* configurations, the topic name is still matched against any topic filters defined in the default QoS settings (e.g., in the **USER_QOS_PROFILES.xml** file). This is new behavior for both *Record* and *Replay*.

4.4.8 Recording Service Administration: 'Database Info' Type Changed

The RTIRecorderDatabaseInfo type in the *Recording Service* administration IDL file (**rtirecord.idl**) has changed; it now uses 64-bit integers for the received and saved byte counts. The new type is defined as follows:

```
struct RTIRecorderDatabaseInfo {
    unsigned long long received_bytes;
    unsigned long long saved_bytes;
};
```

Important: This change introduces an incompatibility between old versions of the IDL file and this one.

4.4.9 Changes in Record and Replay IDL Files

Some changes have been made to the Record and Replay IDL files:

❑ In the Record IDL file (**resource/idl/rtirecord.idl**):

The topic names for administration have been changed so that they are better aligned with other RTI components:

- The command request topic name is now **rti/recorder/administration/command_request**.
- The command request/status topic name is now **rti/recorder/administration/command_response**.
- Some type names and enumeration values in the IDL have been changed so they are more representative.

❑ In the Replay IDL file (**resource/idl/rtireplay.idl**):

The topic names did not have string constants in the IDL file that could be accessed and used by the user. These names have been added to the IDL: **COMMAND_REQUEST_TOPIC_NAME** and **COMMAND_RESPONSE_TOPIC_NAME**.

4.4.10 Association of XML Type Definitions with Topics

This release includes a new configuration tag called **<topics>** within **<type>**. You can use it to make an explicit association between Topics and XML type definitions.

For example:

```
<type>
  <element>
    <register_top_level>true</register_top_level>
      <type_name>CanonicalTypeName</type_name>
      <topics>
        <element>TopicName</element>
      </topics>
   </element>
</type>
```

For more information, see Section 4.10 and Section 7.10 in the *Recording Service User's Manual*.

4.4.11 Ability to Select XML Type to be Registered Based on Topic Name

The *Record* and *Replay* tools now allow you to specify lists of topics associated with the type definitions in the XML files.

A new XML configuration tag, **<topics>**, has been addded to the Type Registration properties in both the *Record* and *Replay* tools. The type configuration settings now have this format:

```
...
<type>
    <element>
        <register_top_level> true </register_top_level>
        <type_name> CanonicalTypeName </type_name>
        <topics>
            <element> TopicName </element>
        </topics>
    </element>
</type>
...
```

You can use the new **<topics>** tag to specify lists of topics names or regular expressions. If these topic names or regular expressions are discovered by the *Record* tool or are about to be replayed by the *Replay* tool, they will use the type definition with the name specified by <type_name>.

4.4.12 Updated to SQLite Version 3.7.17

This release builds against an updated version of SQLite, version 3.7.17. The change log for this version of SQLite is here: http://www.sqlite.org/releaselog/3_7_17.html.

For improved concurrency when accessing the SQLite databases generated by Recorder, SQLite introduced the Write Ahead Logging (WAL) mode in version 3.7.0. WAL is not enabled by default in *Recording Service* but it can be used by setting it with the new pragma execution feature described in Section 4.4.5.

4.4.13 Increased Error Level for 'Incompatible QoS' Error Messages

If an incompatible-QoS error prevented *Recorder Service* from recording data, an error message was shown with 'INFO' level. Now that message will be shown with a higher error level (EXCEPTION).

4.4.14 Increased to Exception Level: All Error Messages that Cause Types not to be Recorded

In previous versions, if *Recording Service* was not able to record data, some error messages were shown with 'warning' level. You might have seen this issue when trying to record deserialized data that did not contain typecode information. The error level for this kind of message has been increased to 'exception' to make it more clear that your data will not be recorded.

4.5 Removed/Deprecated Options

4.5.1 Replay Tool's Command-Line Option '-noAutoEnable' no Longer Available

The *Replay* tool's command-line option, **-noAutoEnable**, has been removed. It cannot be supported because the enable/disable remote administration commands are not supported.

4.5.2 The 'self_contained' Option Deprecated

With the new field selection capabilities, the 'self_contained' option in the Recorder Database properties has been rendered obsolete and starting this version, it is considered deprecated. When the new **<replay_compatibility>** option is active, all the necessary fields in the necessary tables will be propagated to new segments as they are created.

5 What's Fixed in 5.1.0

5.1 Fixes Related to Record Tool

5.1.1 Source and Reception Timestamps Recorded Differently

Although the Sample Info source timestamp and reception timestamp fields were of the same type, the *Record* tool recorded them differently. Source timestamps were stored in two separate numeric columns, while reception timestamps were recorded as text. This problem has been resolved. Now both fields are recorded the same way: one single numeric column representing nanoseconds.

[RTI Issue ID RECORD-43]

5.1.2 Record Tool used Inefficient Database Access to Locate Table

When the *Record* tool discovers a new DataWriter for a topic, it checks for the existence of a table associated with the topic. In the previous release, it checked for the table in an inefficient manner, which could take a long time if the table had a large number of rows. This release uses a more efficient mechanism when searching for the table.

[RTI Issue ID RECORD-387]

5.1.3 Record Tool Locked First Segment in Set Throughout Whole Operation Time

The *Record* tool locked the first file segment recorded in a multi-segment session. This was preventing the user from, for example, moving or deleting it. This problem has been resolved.

[RTI Issue ID RECORD-401]

5.1.4 Wrong Column Limit in Recorded Deserialized Format

The *User's Manual* specified the column limit for recorded data to be 1,950 columns. The tool actually supported just 999 user-data columns. In this release, the column limit has been increased to 5,050 columns.

[RTI Issue ID RECORD-402]

5.1.5 Issues in XSD Definitions for Recorder

Some of the example configuration files for the *Record* tool referred to an incorrect XSD filename, **rtirecorder.xsd**. This caused some valid XML configurations to fail validation. In this release, all example configuration files refer to the correct XSD file, **rti_record.xsd**. The XSD definitions that are used by the configuration can be found in the document **resource/schema/rti_record.xsd**.

[RTI Issue ID RECORD-303]

5.1.6 Recording Failed when <shared_table> Set to True

When a Topic Group in the Record tools's configuration had the <shared_table> tag set to true (which is not the default), the *Record* tool failed and logged the following errors:

```
RTI Recorder started
exception:[RTIDRTUserDataTable_update@678]:near ",": syntax error
exception:[RTIDRTUserDataTable_new@957]:Table update
exception:[RTIDRTUserDataReader_new@896]:Failed to create user subscription
exception:[RTIDRTUserDataSubscriber_create_datareader@276]:Failed to cre-
ate reader
exception:[RTIDRTUserDataSubscriber_add_reader@317]:Failed to create reader
exception:[RTIDRTUserDataEvent_process@164]:Failed to add data reader
```

```
exception:[RTIDRTDataBase_begin_transaction@83]:error locking DB: [cannot
start a transaction within a transaction]
exception:[RTIDRTDataBase_begin_transaction@83]:error locking DB: [cannot
start a transaction within a transaction]
```

The process became unresponsive to normal termination and had to be killed. This problem has been resolved.

[RTI Issue ID RECORD-340]

5.1.7 Problems Recording Multiple Versions of Type in Serialized Format

Recording in serialized format when multiple versions of a type were available (extensible extensibility) was not working properly. The tables may have been mistakenly recorded in deserialized format. This problem has been resolved.

There is a known limitation when DataWriters do not publish typecodes, see Known Issues (Section 6).

[RTI Issue ID RECORD-343]

5.1.8 After Remote Command to Reconfigure with 'auto_start' Set to True, Recording Failed to Start Automatically

When reconfiguring the *Record* tool to use a configuration in which <auto_start> was set to true, recording did not start automatically as expected. This problem hasn been resolved.

[RTI Issue ID RECORD-350]

5.1.9 Unable to Record to Database if File Name Contained non-ASCII UTF-8 Characters (e.g., Arabic)—Windows Platforms Only

On Windows platforms only, *Recording Service* failed to record to a database file if the file's name contained non-ASCII UTF-8 characters, such as Arabic. The database files were created correctly, but no user-data was recorded into them. This problem has been resolved.

[RTI Issue ID RECORD-511]

5.2 Fixes Related to Replay Tool

5.2.1 Ability to Configure Replay to Use a Non-Default Path Separator

SQLite tables created by the recording service have table names constructed from the **topic_name**, **record_group_name**, and **domain_name**, separated by a **path_separator** character. The *Record* and *Replay* tools use the '**$**' character as the default **path_separator** character, but while the *Record* tool could be configured to use a different character, the *Replay* tool could not. This deficiency has been corrected. Now you may specify the **path_separator** character for each **replay_database** entity, using the XML **<path_separator>** tag.

The path separator may be any character except '**!**', '**~**', '****', and '**/**'. (These are the same characters restricted by the *Record* tool). However, when using non-default **path_separators**, you must ensure that the path separator character is unique—i.e., not used in any **topic_name**, **record_group_name**, or **domain_name**. Failure to ensure the uniqueness of the **path_separator** character will most likely cause the *Replay* tool to fail.

[RTI Issue ID RECORD-199]

5.2.2 Replay Wrote Invalid Samples from Database

The *Replay* tool did not take into account the invalid data flag in recorded Sample Info fields and published repeated consecutive samples. This problem has been resolved.

[RTI Issue ID RECORD-290]

5.2.3 Corrupted Typecodes from Replay Service

The *Replay* tool corrupted typecodes sent on the wire when replaying databases stored in serialized mode and when the typecode was taken from the DCPSPublication table. This problem has been resolved.

[RTI Issue ID RECORD-291]

5.2.4 Replaying Same Database File from Multiple Database Entities in Recording Service Failed with "database is locked" Error

Attempting to replay the same database file from multiple entities, with <readonly> set to false and no preexisting indexes in the file, would result in a "database is locked" error and *Replay* would fail. This problem has been resolved.

[RTI Issue ID RECORD-318]

5.2.5 Failure to Replay Topics Recorded with no TypeCode Information in DCPSPublication Table

The *Replay* tool failed to replay data that was recorded without TypeCode information stored in the DCPSPublication table in the database file. Even if the types were provided via XML type configuration, *Replay* was unable to replay and would exit. This problem has been resolved.

[RTI Issue ID RECORD-334]

5.2.6 Dynamic DataReader Failed to Deserialize Mutable Sample Published by Replay Service DataWriter

A Dynamic DataReader failed to deserialize a mutable sample published by a Replay service DataWriter. The issue only occurred when samples were recorded in serialized mode. This problem has been resolved.

[RTI Issue ID RECORD-364]

5.2.7 Indexes not Created in Database Segments

When a new database segment was created, indexes were created for the first segment, but they were not in the subsequent ones. This problem has been resolved.

[RTI Issue ID RECORD-552]

5.3 Fixes Related to Recording Console

5.3.1 Recording Console Log Files not Removed

The log file created by *Recording Console* was not deleted after the application exited. This problem has been resolved; now log files will be removed if no problems were encountered during application execution.

[RTI Issue ID RECORD-365]

5.3.2 Failure to Record or Replay from Console when Type Information Provided via XML

Recording Console could not start the *Record* or *Replay* tools when the input configuration provided the type definitions through XML using the tag <type_config>.

The log files generated by *Recording Console* showed an error like the following:

```
INFO recording service stdout: exception:
[RTIDRT_XmlTypeConfiguration_read_config@864]:Failed to read type configu-
ration 2012-12-13 16:01:57.647 [Thread-16]
```

This problem has been resolved.

[RTI Issue ID RECORD-369]

5.3.3 Slow Start-up for Console

In the previous release, it took a long time for the *Console* to start up. This problem has been resolved.

[RTI Issue ID RECORD-169]

5.3.4 Console Hung While Launching Record or Replay if NDDS_DISCOVERY_PEERS Excluded Local Discovery

The *Console* would hang and display "Launching..." after clicking on the Record or Replay buttons. This occurred on systems configured with the NDDS_DISCOVERY_PEERS environment variable set in a way that prevented local discovery (e.g., not including shared memory, and not including localhost or any other such option), thus affecting the QoS used for communication between the *Console* and the *Record* or *Replay* tools, preventing them from discovering each other.

The *Console*, *Record*, and *Replay* tools always run on the same machine. The *Console* launches and configures the *Record* or *Replay* tools, and in particular, configures the communication QoS between the tools and the *Console*.

The configuration provided by the *Console* in the previous release did not account for the case mentioned above. In this release, the administration domain used by the *Console* and the *Record/Replay* tools to intercommunicate is explicitly configured by the *Console* to set 'initial_peers' to shared memory, thus, overriding and preventing any other value for this attribute (whether by a configuration file or environment variable).

This change will only affect the domain used by the *Console* and the *Record/Replay* tools to communicate (domain 99 by default); it will have no effect on any other domain that the service connects to.

[RTI Issue ID RECORD-344]

5.3.5 Issues with Synchronized Control and State Indication between Console and System Tray Icon

When controlling *Administration Console* from the system tray icon (with a right click), or when the console would perform a scheduled operation, the state indication in the console would not update appropriately. For instance, when a scheduled replay operation would begin, the UI LCD would turn green, but the replay button would not light up (since it was not clicked on).

This release improves the state indication to indicate the state in the same way, no matter what triggers the console state change. Such triggers include controlling the console from the system tray icon (right click), the scheduler, or even by changing the service state of the service associated with the console by the *Administration Console*.

[RTI Issue IDs RECORD-323, RECORD-343, RECORD-352]

5.3.6 No Indication of Why Replay was Disabled when Recording File was Empty

When loading an empty recording file, there was no indication of why the Replay button was disabled on the *Console*.

This problem has been resolved. If the loaded recording file is empty, Replay will be disabled and the settings panel will show that the file is empty.

[RTI Issue ID RECORD-335]

5.3.7 Improved Response Time for Opening Settings Panel

In previous releases, clicking on the button to open the settings panel would not respond instantaneously if *Recording Console* had been previously set to be configured with a file. Responsiveness has been improved in this release.

[RTI Issue ID RECORD-356]

5.3.8 Crash in Recording Console when Changing Replay or Recorder Domain ID while Changing Views

When changing the Replay or Record domain ID in *Recording Console*, and quickly switching to another panel (such as the Topics panel), the application could have crashed if the domain ID change didn't complete before changing. This problem has been resolved.

[RTI Issue ID RECORD-403]

5.3.9 Possible Recording Console Frozen Status

When recording using *Recording Console* and loading an external configuration, you may have seen *Recording Console* not update its status (staying in the "launching..." state). This was due to a missing administration domain ID in the XML configuration. Now *Recording Console* captures this case and adds a default administration domain ID.

[RTI Issue ID RECORD-562]

5.3.10 Recording Console Appeared Unresponsive if Deprecated Tag used in Configuration File

You may have seen *Recording Console* remain in the "launching" state after it started recording with an external configuration file. This was due to the use of a XML property that is no longer available. This problem has been resolved.

[RTI Issue ID RECORD-563]

5.3.11 Error from Recording Console when Using domain_type_config Property

You may have seen an error when loading a recorder configuration file containing the property **domain_type_config**. This was due to an error in the XSD file used by *Recording Console* when parsing the type-definition-file name. This generated an exception that was found in the *Recording Console* log. This problem has been resolved.

[RTI Issue ID RECORD-554]

5.4 Fixes Related to Converter

5.4.1 Converter was Missing Headers for Typedef Fields

There was a problem when using *Converter* (**rtirecconv**) with types that contained typedefs. The converted files had missing headers for the field names. This problem has been resolved.

[RTI Issue IDs RECORD-192 and RECORD-305]

5.4.2 Converter Failed to Properly Convert Databases Containing Types with Compact Arrays

Converter failed when converting tables for types that contained compact arrays of bytes or characters. This problem has been resolved.

[RTI Issue ID RECORD-197]

5.4.3 Segmentation Fault when Converting Very Long Double or Float Values

Converter issued a segmentation fault when converting very large real values, e.g. -1.0e310. This problem was due to the conversion format used to convert the number. This format is now conditional to the size of the resulting number. When the converted string is larger than the maximum, the number will be converted using scientific notation.

[RTI Issue ID RECORD-537]

5.4.4 Converter Failed to Work Properly with Wide Strings

Converter was not working properly with wide string types, resulting in incomplete strings in the converted files or even exceptions in the serialized case. This problem has been resolved.

[RTI Issue ID RECORD-542]

6 Known Issues

6.1 Issues Related to Replay Tool

❑ The *Replay* tool currently does not support the following XML configuration modes:

- <replay_service> <auto_exit> (has no effect)
- <replay_topic> <output> <keyed> (has no effect)
- <time_control> <start_mode> MATCHED or LOOP modes
- <time_control> <rate> AS_FAST_AS_POSSIBLE (except for session level)
- <topic_time_control> <start_mode> MATCHED mode

❑ Limitations with the *Replay* tool's shell commands:

- The **step** command is functional for session and topic entities only (not service or database)
- The **rate** command is functional for topic entities only

❑ Performance and indexing with the *Replay* tool:

The *Replay* tool replays stored samples in the same order in which they were received, using SQLite indexes to retrieve the samples in sorted order. SQLite automatically builds indexes when opening an SQLite table for sorted access; for large tables the process of building the index may take some time. To improve *initialization* performance, the *Replay* tool attempts to create and store indexes, rather than depend upon automatic indexing, for the tables which it will be replaying, saving initialization time on subsequent replays.

The *Replay* tool's ability to store indices is controlled by the <readonly> parameter of the <replay_database>. Setting <readonly> to true prevents *Replay* from storing indices for a table; in this mode. the *Replay* tool will display a message during initialization for each table opened stating that it was unable to store the table index. Setting <readonly> to false (the default) will allow the *Replay* tool to write the table indices to the database.

The *Replay* tool's performance is not affected by this option; it will use the fastest means of retrieving samples in either case. But setting the <readonly> option to false may help improve the tool's *initialization* performance.

❑ When loading a large file for playback, please be aware that this operation may take some time.

❑ If you load the configuration file, **examples/replay_simple_config.xml**, and select the **fast_replay** configuration profile while using your own recorded data file (instead of the example recording from RTI), the *Replay* service will exit and log a message regarding 'no match in the recording for A_Topic.'

- ❏ The *Record* and *Replay* Shells are not completely compatible with standard input piping of commands.
- ❏ For *RTI Admin Console* to work properly with the *Replay* tool, do not use the XML <name> tag under <administration>. *Admin Console* will not recognize the replay service and will not be able to administer it. This will be addressed in a future release. [RTI Issue ID BIGPINE-429]

6.2 Issues Related to Recording Console

- ❏ In *Recording Console*, when changing playback speed, or skipping to another playback location, occasionally playback will appear stuck (it is actually paused). The workaround is to click the Pause button twice.
- ❏ *Recording Console* may fail to shut down gracefully after stepping through to the end of a recording. If a recording is paused and then stepped through to the end, the *Replay* service may not shut down properly. In this case, *Recording Console* displays an error that the service stopped unexpectedly. [RTI Issue ID RECORD-135]
- ❏ Interaction between *Recording Console* and *Admin Console*

  This issue only applies if you are using *Recording Console* and *RTI Admin Console* at the same time, and you have configured *Admin Console* to join domain ID 99. In this scenario, do not use *Admin Console* to pause or disable any *Recording Console* services (their names begin with "RTI-Recorder-" or "RTI-Replay-"). Doing so may cause an error in *Recording Console*. [RTI Issue ID BIGPINE-795]
- ❏ *Recording Console* will not reflect stopped status if recording is stopped by another tool.

  When recording data with *Recording Console*, *RTI Admin Console* can send a command to stop the recording. In this case, recording will stop but *Recording Console* won't reflect the stopped status in any way; it will appear that recording is still in progress, although the file won't grow in size.

  Pause commands work fine and are reflected by both sides, *Recording Console* and *Admin Console*.

  [RTI Issue ID RECORD-253]
- ❏ Welcome screen may appear blank on some platforms

  The welcome screen may appear blank if the operating system does not have a web browser that is compatible with Eclipse. [RTI Issue ID DIABLO-538]

6.3 Issues Related to Converter

- ❏ When using *Converter* on a recording created with *Recording Console*, you may see a warning related to internal topics used by the *Console*:

  ```
  exception:[RTIConverterModelPublisherCallback@2293]:Failed to create type
  com_rti_tools_remotectx
  ```

  You can safely ignore the warning—the conversion results *are* valid.
- ❏ *Converter* (**rtirecconv**) cannot convert tables with only a subset of the data. In general, if you record in deserialized mode, use the **sqlite3** command to convert to HTML and CSV; if you record in serialized mode, use *Converter*.
- ❏ In files recorded on Windows systems, the recorded timestamp is the number of microseconds since the device was booted, not since January 1, 1970. Therefore the **-time gmt** option to *Converter* (**rtirecconv**) will not show the correct time.

6.4 Other Issues

- When you record a database using the PRAGMA feature (**<sqlite_pragmas>** in the **<recorder_database>** settings), the resulting databases may be incompatible with *Recording Console*. This is due to a third-party incompatibility. The following exception will be thrown:

```
java.sql.SQLException: file is encrypted or is not a database
```

  To replay the database, use the *Replay* tool.

  [RTI Issue ID RECORD-574]

- Recording and/or replaying mutable types requires the type definition to be provided via XML configuration using the <type_config> tag. If the type definition is not provided via XML, the *Record* tool will display the following error messages:
  - When recording in deserialized mode:

    ```
    Failed to get valid typecode information for Publisher. Recorder cannot
    confirm that the entity publishes a supported type.
    ```

  - When recording in serialized mode:

    ```
    DDS_DynamicData_from_stream:ERROR:Bad
    parameter:encapsulation_kind of stream
    ```

- To record a data type that has more than 5,050 primitive types, you must set the **deserialize_mode** property to RTIDDS_DESERIALIZEMODE_NEVER. Otherwise, you will see the following error message and recording will fail:

  ```
  "exception:[RTIDRTUserDataTable_update@610]:too many SQL variables"
  ```

  [RTI Issue ID RECORD-38]

- The DynamicData API does not support out-of-order assignment of members with a length greater than 65,535 bytes. In this situation, the following error is reported:

  ```
  sparsely stored member exceeds 65535 bytes
  ```

  For example:

  ```
  struct MyStruct {
      string<131072> m1;
      string<131072> m2;
  };
  ```

  With the above type, the following sequence of operations will fail because **m2** is assigned before **m1** and has a length greater than 65535 characters.

  ```
  str = DDS_String_alloc(131072);
  memset(str, 'x', 131072);
  str[131071]= 0;
  DDS_DynamicData_set_string(
      data,"m2", DDS_DYNAMIC_DATA_MEMBER_ID_UNSPECIFIED, str);
  DDS_DynamicData_set_string(
      data,"m1", DDS_DYNAMIC_DATA_MEMBER_ID_UNSPECIFIED, str);
  ```

  If the member **m1** is assigned before **m2,** the sequence of operations will succeed.

  [RTI Issue ID CORE-3791]

- RTI does not recommend using files that are mounted over NFS to store recorded data. *Recording Service* uses file-locking, which has known issues working over NFS. If file-locking is not working, *Recording Service* will hang. In particular, this problem may appear on Yellow Dog Linux systems.

- Leading and trailing spaces in a Topic Name are ignored. However, spaces within the string are allowed. For example, " My Topic " will be treated as "My Topic".

- ❏ Fully qualified field names in struct's cannot be longer than 1,024 characters.
- ❏ Sequence and array indices cannot be used in Topic or Field expressions.
- ❏ *Recording Service* cannot communicate with DataReaders or DataWriters of Topics with a data type that includes bit fields. You may see the following message, but *Recording Service* will continue to work normally otherwise:

  ```
  DDS_DynamicDataTypeSupport_initialize:type not supported (bitfield member)
  ```

  [RTI Issue ID CORE-3949]
- ❏ *Recording Service* and *Converter* cannot deserialize bit fields. If this type is used, the deserialize mode must be RTIDDS_DESERIALIZEMODE_NEVER.
- ❏ If the *Connext* application being recorded has a keyed data-type and **DataWriterProtocolQosPolicy.disable_inline_keyhash** is set to TRUE (not the default), *Recording Service* may misinterpret samples as being from the wrong instance.
- ❏ If you start an instance of the *Record* tool using command-line options (not a configuration file), then sending a new configuration to that instance of the *Record* tool using the remote shell will not work.
- ❏ When <time_mode> is set to TOPIC_RELATIVE, the first sample in a recording is not sent right away when replay starts. [RTI Issue ID RECORD-133].
- ❏ There is a known limitation when recording data in serialized format in environments where multiple versions of a type are published. If the writers do not publish their typecode information, the *Record* tool may store samples from unwanted versions. [RTI Issue ID RECORD-346]
- ❏ On Windows 8 systems, be aware of a limitation in the OS regarding the write permissions in some folders. Even if you are using an administrator account, some folders (such as C: or "Program Files") cannot be used to store user data. If you try to create a recording database there, Windows 8 will automatically create it in a virtual storage unit (usually found under **C:\Users\<*user_name*>\AppData\Local\VisualStore**). This folder might be hidden. [RTI Issue ID RECORD-525]