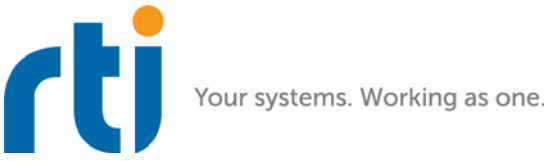
RTI Recording Service

Release Notes

Version 5.3.1



Printed in U.S.A. First printing.
February 2018.

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:

- All Linux® platforms on x86 and x64 CPUs listed in the *RTI Connext® DDS Core Libraries Release Notes* with the same version number, except Wind River® Linux and any custom target Linux platform.
- Raspbian Wheezy 7.0
- All OS X® platforms in the *RTI Connext DDS Core Libraries Release Notes* with the same version number.
- QNX® 6.6 custom target platform on x86 (i86QNX6.6qcc_cpp4.7.3).
- All Windows® platforms in the *RTI Connext DDS Core Libraries Release Notes* with the same version number.

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

2 Compatibility with Other RTI Products

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

Recording Service 5.1.0 and higher is compatible with *RTI Connext DDS* 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 and higher is not backwards compatible with databases recorded with previous releases of *RTI Recording Service* (5.0 and lower). 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 Connext DDS Core Libraries Release Notes*.
- In *Connext DDS* 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,

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

Recording Service 5.1.0 and higher is not out-of-the-box compatible with applications running older versions of *Connext DDS* or *RTI Data Distribution Servic*e. Please see the *RTI Core Libraries Release Notes* for instructions on how to resolve this compatibility issue with older *Connext DDS* and *RTI Data Distribution Service* applications.

- Some changes were made to the *Record* and *Replay* IDL files starting in *Connext DDS* 5.1.0:
  - In the *Record* IDL file (**resource/idl/rtirecord.idl**):

    The topic names for administration were changed for better alignment 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**):

    Topic names did not have string constants in the IDL file that you could use. These names have been added to the IDL: COMMAND_REQUEST_TOPIC_NAME and COMMAND_RESPONSE_TOPIC_NAME.
- 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 and higher.
  - The *Record* and *Replay* shells, *Admin Console,* and *Connext* 5.1.0 and higher user applications performing monitoring/administration are not compatible with *Recording Service* 5.0.0.

2.1 Command-Line Options Compatibility

Starting with 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, *Recording Service Integration with Extensible Types*, in the *Recording Service User's Manual*.

---

3 What's New in 5.3.1

This section highlights improvements in 5.3.1. These enhancements have been made since 5.3.0.

3.1 Ability to Record Distributed Logger's LogMessage Topic

Recording Service discarded, by default, all of Distributed Logger's topics/types automatically. This was done by design, because *Recording Service's* XML design allows you to use the same DomainParticipant for administration/distributed logger and user-data. However, this could produce type-registration issues when attempting to record Distributed Logger messages in the domain used by administration. This same problem applies to the Monitoring topics; they cannot be recorded in a domain where monitoring has been enabled, because of type-registration conflicts.

This release removes this limitation, so now it is possible to record Distributed Logger's LogMessage topic (rti/distlog).

Note: See Recorded Discovery Information Missing after Stopping and Restarting Recorder (Section 4.12). This fixed bug offers a mode in which the administration domain is completely separate from the user-data domains. This relates to this issue, providing a way to safely record Distributed Logger in the user-data domains while being able to enable Recorder's Distributed Logger in the administration domain by using a dedicated DomainParticipant.

4 What's Fixed in 5.3.1

This section describes bugs fixed in 5.3.1. These fixes have been made since 5.3.0.

4.1 Recording Console Failed When Trying to Parse XML Configuration File Containing Environment Variable References

Recording Console failed when parsing an XML configuration file that contained environment variable references, as shown in the following example:

```
...
    <domain name="domain0">
        <domain_id>$(DDS_DOMAIN)</domain_id>
    </domain>
...
```

This issue affected only *Recording Console*. The *Record* tool works fine with the above example. This problem with *Recording Console* has been resolved.

[RTI Issue ID RECORD-856]

4.2 Replay did not Check Existence of Column

Replay may have failed in some situations when checking for the existence of a column. This operation is necessary when initializing the tool. For example, if a table was present but had no rows in it, *Replay* failed with the following error.

```
RTI Replay started
PLAYBACKTopic_SQLiteColumnExists:!Unexpected SQLite status for SQL
statement step.
```

This problem has been resolved.

[RTI Issue ID RECORD-871]

4.3 Recorder, Converter, and Replay had Incorrect Column Limits Set for SQLite

The *Record* tool, *Converter,* and *Replay* were incorrectly configured to use the SQLite default of 2,000 columns instead of the specified 5,000. This could cause problems, for example, when recording types with more than 2,000 columns in deserialized format or when using *Converter* to deserialize a serialized table that would take more than 2,000 columns.

You may have see error messages like the following:

```
SQLite error: too many columns on <table name>
```

Replay may also have failed to replay a database with more than 999 columns, due to the parameter binding limitation of SQLite being set to its default value, and not 5,000.

This problem has been resolved. The column and parameter binding limits have been restored to 5,000.

[RTI Issue ID RECORD-874]

4.4 Crash when Converting Type Containing Very Long Octet Sequence

Converter may have crashed when converting types containing a very large octet sequence. For example:

```
const long MAX_BUFFER_SIZE = 3000000;
struct test {
    sequence <octet, MAX_BUFFER_SIZE> x;
};
```

This problem has been resolved.

[RTI Issue ID RECORD-875]

4.5 Memory Leak in Replay when Writing Serialized Samples

There was a memory leak in the *Replay* tool when writing a serialized sample. The memory lost was proportional to the size of the written sample. This problem has been resolved.

[RTI Issue ID RECORD-876]

4.6 Recorder Crashed upon Receipt of START Command when in STOPPED Mode

When started in IDLE mode, or when stopped and then restarted, the *Record* tool crashed due to a segmentation fault in an SQLite call with a null parameter. This problem has been resolved.

[RTI Issue ID RECORD-878]

4.7 Recorder Failed to Update Verbosity with Command-Line Parameter

In release 5.3.0, using the **-verbosity** command-line parameter had no effect. The requested verbosity was not used. This problem has been resolved.

[RTI Issue ID RECORD-879]

4.8 Conversion to CSV Resulted in Incorrect CSV File

When using the option to convert to CSV, *Converter* in version 5.3.0 generated an incorrect CSV file:

- The topic name normally exported was missing.
- There were misplaced new lines that caused the file to be imported incorrectly by tools such as Excel®.
- When string fields contained new lines, they were ignored and did not appear in the output.

This problem has been resolved. This release also adds compliance with the RFC-4180 standard for CSV files regarding string fields being exported, thus allowing strings with new lines, commas, or double quotes to be exported and imported correctly.

[RTI Issue ID RECORD-887]

4.9 Converter Failed when Working with Tables Recorded in Old Serialized Format

There are two cases in which the *Record* tool uses the old serialization format to record serialized data:

- When using the <fast_serialized_mode> option.
- When the *Record* tool cannot find a type representation for the topic, via discovery or XML.

In these cases, the format used to record the data is the RAW_BYTES type. A flag is added to the log entry that is created when the table is created.

Converter failed to detect the serialization format flag and apply it properly when reading the serialized samples, using the wrong DynamicData API. This problem has been resolved.

[RTI Issue ID RECORD-892]

4.10 Converter Failed when Deserializing Keyed Topic Table with Invalid Data Samples

When the *Record* tool records invalid data samples (samples with valid_data = 0), for example, a dispose message or a not-alive-no-writers message, it stores **null** as the data sample in serialized format.

These samples caused *Converter* to fail with the following error:

```
Deserializing table: Circle$RecordAll$domain0...[1/5]
DDS_DynamicData_from_cdr_buffer:ERROR: Bad parameter: buffer
exception:[RTIConverterModel_processSerializedToDeserialized@4723]:Failed
to deserialize recorded serialized sample
exception:[RTIConverterModel_deserializeTable@5354]:Failed to process seri-
alized table
exception:[RTIConverterModel_serializedToDeserialized@5634]:Failed to pro-
cess table Circle$RecordAll$domain0
exception:[RTIConverterModel_serializedToDeserialized@5681]:Failed to close
database handle; SQLite error: unable to close due to unfinalized state-
ments or unfinished backups
exception:[main@672]:failed to convert <<file>>
RTI Recording Service - Convert: Finished
```

This problem has been resolved.

[RTI Issue ID RECORD-901]

4.11 Converter Failed with Types Containing Fields Larger than 65,535 Bytes

When running *Converter* with types containing any field larger than 65,535 bytes, you may have seen the following error because a limit in the DynamicData API was reached:

```
DDS_DynamicData_unbind_complex_member:internal error 1 trying to assert
complex member
exception:[DRT_DynamicType_get_matching_fields_internal@2298]:Failed to
unbind complex member.
DDS_DynamicData_finalize:WARNING: destructing object bound to a member,
automatically unbinding now
DDS_DynamicDataStream_assert_complex_member:!sparsely stored member exceeds
65535 bytes
DDS_DynamicData_unbind_complex_member:internal error 1 trying to assert
complex member
exception:[DRT_DynamicType_get_matching_fields_internal@2307]:Failed to get
matching fields from a sample's complex member.
```

Converter was using the DynamicData API to make the field calculations. To resolve this problem, Converter now uses the TypeCode API, so the DynamicData limit is no longer an issue.

[RTI Issue ID RECORD-902]

4.12 Recorded Discovery Information Missing after Stopping and Restarting Recorder

After recording was stopped (via a remote STOP command) and restarted (via a START command), discovery information may have been missing from the discovery tables (DCPSPublication, DCPSSubscription and DCPSParticipant). *Recording Service* did not tear down the participants; thus discovery did not re-occur, so this information was not recorded.

This problem has been resolved. Now *Recording Service* will tear down all user-defined DomainParticipants, except the one used for administration. This means that, by default, rediscovery related to the administration domain still won't occur, and some discovery data may still be missing. To address this, we have added new behavior to the <remote_access_domain> configuration tag. If a domain ID is specified (instead of a reference to a user-defined domain), the DomainParticipant used for administration will be created separately from the pool of user-defined domains. In this case, all the user-defined domains will be destroyed when *Recording Service* is stopped, thus all the entities will be rediscovered when a START command is received.

Note: In summary, to completely avoid this issue, there are two options:

- Change the <remote_access_domain> tag so it explicitly declares a domain ID and doesn't reference any of the user-defined domains.
- Or, if you still want to define the administration domain by referencing a user-defined domain, make sure that domain is not used for recording user data at all (that is, do not associate the domain with any record_group).

[RTI Issue ID RECORD-907]

4.13 Converter Exported Some Union Field Values as NULL Instead of Their Actual Values

Converter exported some union fields as NULL instead of using the values associated with their discriminator values. This problem has been resolved.

[RTI Issue ID RECORDING-910]

4.14 Converter Failed to Export Deserialized Table Containing Type with Primitive Aliases

Converter could fail to export a database recorded in deserialized format where the type contained aliases of primitive types. For example:

```
typedef char Chart_T;
...
struct Type {
    Char_T char_field;
    ...
};
```

This issue has been resolved.

[RTI Issue ID RECORD-917]

4.15 Potential Segmentation Fault in Converter when Swapping to the Next File Segment while Deserialising

Converter could potentially cause a segmentation fault when deserializing serialized tables, at the point where a new deserialized file segment was created. This issue has been resolved.

[RTI Issue ID RECORD-921]

4.16 Recorder Failed during Rollover of Log File for Large Database Names

Recorder failed when trying to roll over (change the segment of) the log file. The failure happened with very large database names (with file paths having more than 224 characters). For these long file names, *Recorder* stopped writing anything in the log file after the rollover.

The console output from *Recorder* (where the error appeared) was:

```
RTI Recorder started
DL Info: RTI_DL_XMLConfig_startWithXMLConfiguration: DL not enabled in XML
Options. It won't be created
Recording to file <very long path exceeding 224 bytes>
exception:[RTIDRTManagerDb_copyDiscoveryTables@773]:Could not create ATTACH
SQL command text
exception:[RTIDRTManagerDb_checkSegmentSizeAndSwapIfFull@1003]:Failed to
copy old Participant, Publication and log tables to new database file
exception:[RTIDRTManagerDb_checkSegmentSizeAndSwapIfFull@1026]:Recording
stopped, error checking size and/or changing segments
exception:[RTIDRTManagerEvent_processDB@299]:Error checking database seg-
ment size
```

This problem has been resolved.

[RTI Issue ID RECORD-922]

4.17 Converter Failed to Convert Deserialized Table Containing Type with Inheritance

When attempting to convert a table in deserialized format that stored a type that inherited from another type (for example, from the Shape Extended type), *Converter* would fail to convert. It would give the following error:

```
Running RTI Recording Service - Convert on rti_recorder_default.dat_0_0
Converting to XML
Exporting discovery tables...
Exporting user-data tables...
Exporting table: Circle$RecordAll$domain0...
DDS_DynamicData_set_long:type mismatch for field color (id=1)
exception:[DRT_DynamicData_set_data_value@5620]:Internal process to set
data value to DDS_DynamicData instance.
exception:[DRT_DynamicData_set_field_value@5762]:Failed to set a value in a
field.
exception:[RTIConverterModel_copyFromFields@2515]:DynamicData set failure
exception:[RTIConverterModel_userDataTableCallback@3076]:Failed to serial-
ize user-data fields
exception:[RTIConverterModel_convert@6425]:Failed to convert table: Circle
exception:[main@649]:failed to convert rti_recorder_default.dat_0_0 to XML
RTI Recording Service - Convert: Finished
```

This issue did not happen when the table was stored in serialized format.

This issue has been resolved.

[RTI Issue ID RECORD-923]

4.18 Replay's start_offset did not Work Correctly

The 'start_offset' field in *Replay* did not work properly. When the following command was used to start the service:

```
rtireplay -cfgFile replay_config.xml -cfgName default
```

The service produced the following output:

```
RTI Replay 5.3.0 initializing ...
RTI Replay initialized
Pausing 30 seconds for discovery...
[Square]: Topic started
RTI Replay started
```

Replay, however, did not actually pause 30 seconds, but started replaying almost immediately. (The replay start was 50 times faster than it was supposed to be, so most of the time it looked immediate.)

This problem has been resolved.

[RTI Issue ID RECORD-924]

---

5 Previous Release

This section includes:

5.1 What's New in 5.3.0

This section highlights new platforms and improvements in 5.3.0. These enhancements have been made since 5.2.3.

5.1.1 New Platforms

This release adds support for these platforms:

- Mac OS X 10.12
- Red Hat Enterprise Linux 6.8
- Ubuntu 16.04 LTS
- Windows Server 2016

For details on these platforms, see the R*TI Connext DDS Core Libraries Platform Notes*.

5.1.2 Replay now Able to Publish Data in Original Partition in which it was Received

The *Replay* tool is now able to read the original partitions in which publishers and associated writers were publishing when the *Record* tool discovered them, and automatically apply them to *Replay's* Publishers and writers. *Replay* will also load any partition changes that occurred during the specified start and stop times and apply them to the DDS entities. See Section 7.11, *Using the Recorded System's Original Partition QoS Information*, in the *Recording Service User's Manual*.

5.1.3 Recording Service now Stores Type-Objects as Well as Type-Codes

Previous versions of *Recording Service* worked only with type-codes in serialized format when using types via discovery. Now *Recording Service* will first look for a type-object, then look for a type-code. The type-object will be serialized and stored in the DCPSPublication table.

This allows the *Record* tool, *Replay,* and *Converter* to work with mutable types and types containing optional members without the need to provide type information via XML, as in previous releases.

5.1.4 Record and Replay Tools now Set Service QoS Policy

The *Record* and *Replay* tools both use the Service QoS policy, by setting the service kind to DDS_RECORDING_SERVICE_QOS and DDS_REPLAY_SERVICE_QOS, respectively.

5.1.5 Record, Replay, and Converter no Longer Directly Depend on OpenSSL when Security Features not Used

The Security Plugins host bundle includes a new library (**rtirecordsec.dll** on Windows systems, **librtirecordsec.so** on Linux systems, **librtirecordsec.dylib** on OS X systems).

Starting in release 5.3.0, this library is only loaded if security features are in use. This library is the only integration point with OpenSSL.

5.1.6 Improved Performance when Recording Final and Extensible Types

The DynamicData API used by the *Record* tool to obtain the buffer to be stored for a sample in serialized format was changed in 5.2.3 to better integrate with X-Types. However, using this API was not necessary in all cases; it is only needed when types contain advanced X-Types features, such as mutability or optional members. This API call is slower than the one previously used.

In this release, when recording types that are final or extensible (not mutable or containing optional members), the *Record* tool will use the older, quicker API call to store the serialized buffer in the database, like it did in 5.2.0.

Moreover, there is a new option for serialized recording called <fast_serialized_mode> in XML. When enabled, it can significantly increase performance when storing final and extensible types in serialized for-

mat. It cannot be used with mutable types or types with optional members. The new tag is located under the <domain> tag.

When using the new serialized mode, take into account that X-Types type coercion is automatically disabled and type matching uses the old method for type matching, based just on type name comparison. So in order for the *Record* tool to record the types, type-name matching must be ensured. Because of this incompatibility with X-Types, the <fast_serialized_mode> option is disabled by default.

5.1.7 Error when Using Field Filters and No Field in Topic's Type Matched

If you ran the *Record* tool using field filters (with <field_expr> under <topic_group>), but no field in the topic's type matched the filter expression, you may have seen this error:

```
RTI Recorder started
Recording to file remote_example.dat_1_0
exception:[RTIDRTUserDataTable_create@589]:Table name: test.RTIDDS_SUB-
SCRIPTIONS
$RecordAll$domain1, CREATE SQL statement failed; Error: near ")": syntax
error
exception:[RTIDRTUserDataTable_new@1641]:Failed to create table
test.RTIDDS_SUBS
CRIPTIONS$RecordAll.domain1
exception:[RTIDRTUserDataReader_new@516]:Failed to create DRT user-data
table object; table name: test.RTIDDS_SUBSCRIPTIONS$RecordAll.domain1
```

This situation caused the *Record* tool to create an incorrect SQL statement and fail to create a table for the topic. This error condition will now be detected and a proper, more informative error message will appear.

5.1.8 Support for Native Heap Monitoring

Recording Service incorporates a native heap memory monitor that allows you to analyze the allocations performed at the service and *RTI Connext DDS* layers. You can use heap monitoring through the command line with the following options:

-heapSnapshotPeriod: <sec> Enables heap monitoring. Generate heap snapshot every <sec>.

-heapSnapshotDir: <dir>> Output directory where the heap monitoring snapshot are dumped. The filenames of the generated dump files have the following format:

RTI_heap_<appName>_<processId>_<index>.log

where <appName> is the name you assigned to the service execution through the **-appName** parameter, <processId> is the process ID of the service execution, and <index> is an integer that automatically increases each snapshot period.

For details related to the format of the snapshot files see the API Reference HTML documentation for *Connext DDS*.

5.1.9 Recording Console's Underlying Eclipse Version Upgraded

Recording Console now builds against Eclipse Neon starting with this release.

5.2 What's Fixed in 5.3.0

This section describes bugs fixed in 5.3.0. These fixes have been made since 5.2.3.

5.2.1 QoS Topic Filters did not Work with Profile Inheritance

When using release 5.1.0, if you used Topic Filters in the XML QoS configuration for the *Record* or *Replay* tools, the filters did not correctly inherit from a base profile. This meant that you needed to copy all the QoS settings from a base profile if you wanted to use Topic Filters. This has been resolved. Topic filters will now inherit correctly from the base QoS profiles.

For details on using Topic Filters, see Section 17.3.4 in the *RTI Connext DDS Core Libraries User's Manual*.

[RTI Issue ID RECORD-611]

5.2.2 Recorded Dispose/Unregister Messages for Keyed Topics were not Replayed

The *Replay* tool ignored recorded 'dispose' and 'unregister' messages in keyed topic tables. In this release, *Replay* will replay these messages. When *Replay* finds a 'dispose' message in the database, it will dispose the associated instance; when it finds a 'not-alive-no-writers' message, *Replay* will unregister the instance.

Users should take into account that the *Record* tool only records the instance state flow that was provided by the middleware, which in some cases may not be the full information. *Replay* will behave as well as possible with the recorded information at hand.

When recording keyed topics, by default the *Record* tool will now record two SampleInfo fields that weren't recorded by default in previous releases: **SampleInfo_instance_handle** and **SampleInfo_instance_state**. *Replay* uses this information to reproduce the instance state.

[RTI Issue ID RECORD-668]

5.2.3 Databases Recorded with auto_start = False had No Discovery Data in DCPSPublication Table

When the *Record* tool was run with the auto_start option set to false, the DCPSPublication table would miss the discovery information for the discovery process that happened until the tool started recording.

The *Record* tool has now three start modes, which are defined in a new XML tag called auto_start_mode (which substitutes and deprecates the old auto_start XML tag):

- RECORD_ENABLED: this mode is equivalent to the old auto_start flag being set. The *Record* tool will start recording discovery data and user data automatically as soon as possible.
- DISCOVERY_RECORD_ENABLED: in this mode, the *Record* tool will start recording discovery data automatically, but it will not record any user data until a start command is received.
- RECORD_DISABLED: this mode is equivalent to the old auto_start flag being unset. The *Record* tool will not record any data, discovery or user data, until a start command is received.

If you are going to use *Replay* or *Converter*, but you don't want to start recording user data right away, the DISCOVERY_RECORD_ENABLED is the best option, so *Replay* and *Converter* will have available discovery information in the DCPSPublication table.

[RTI Issue ID RECORD-699]

5.2.4 Converter Failed to Export Data Containing Unions with Default Members

Converter failed when trying to convert to CSV format if the type contained unions with default members and there were samples in the database that did not use the default member. The process would fail with the following error:

```
> rtirecconv -format CSV -decodeChar text -decodeOctet hex -time epoch -
compact auto <database file>
Running RTI Recording Service - Convert on reproducer
Converting to CSV
DDS_DynamicData_get_ulong:deserialization error: unsigned long
exception:[DRT_DynamicData_print_primitive_field@4915]:Failed to get data
value from sample.
exception:[DRT_DynamicData_print_union_field@5628]:Failed to print primi-
tive field.
exception:[DRT_DynamicData_print_union_field@5679]:print union_field failed
exception:[DRT_DynamicData_print_struct_field@5869]:print struct_field
failed
```

```
exception:[DRT_DynamicData_print_sequence_field@5454]:Internal process to
print sample failed.
exception:[DRT_DynamicData_print_sequence_field@5506]:print sequence_field
failed
exception:[DRT_DynamicData_print_struct_field@5869]:print struct_field
failed
exception:[RTIConverterModel_userDataTableCallback@1317]:failed to iterate
over sample
exception:[RTIConverterModel_convert@3584]:Failed to convert table:
ice::EnumerationSettContext
exception:[main@574]:failed to convert reproducer to CSV
RTI Recording Service - Convert: Finished
```

This problem has been resolved.

[RTI Issue ID RECORD-707]

5.2.5 Could not Replay Database File with Non-Zero Set Number

When using version 5.2.3, if you attempted to load any database file of the usual form <filename>[name].dat [set][segment]</filename> and the segment number was non-zero, you would get the following error and *Replay* would stop:

```
PLAYBACKConnectionPool_exe: Failed to execute SQLite statement SELECT
MIN(Timestamp) FROM RTILog
PLAYBACKDatabase_SQLiteGetDatabaseTimestamp SQLite error: out of memory
```

This problem has been resolved.

[RTI Issue ID RECORD-782]

5.2.6 Converter 5.2.3 was not Backward-Compatible with 5.2.0 Databases in Serialized Format

For better integration with X-Types, the serialization mechanism was changed in release 5.2.3. However, *Converter* did not support the new format. This caused databases created with 5.2.0 not to be loaded properly. This problem has been resolved.

[RTI Issue ID RECORD-791]

5.2.7 Disposed Instance Triggered Serialization Error in Recorder

When the *Record* tool was configured to store samples in serialized format and it received a propagated dispose message, this triggered an error:

```
DDS_DynamicData_to_cdr_buffer:serialization error: buffer exception: [Meta-
dataFieldModel_bindUserDataFields@1346]:Failed to get CDR buffer for sam-
ple; cannot add sample to database in table
Test$Record$Domainexception:[RTIDRTUserDataReader_storeData@829]:Failed to
bind data for field USER_DATA in table Test$Record$Domain
```

The error occurred because the *Record* tool tried to add the dispose message to the database.

The *Replay* tool may also have failed when attempting to replay the sample.

This problem has been resolved. Now we bind a serialized dispose message with a null value and no length in the database.

[RTI Issue ID RECORD-792]

5.2.8 Replaying Serialized Keyed Data used Wrong Instance Handle

Replaying a database stored in serialized format may have resulted in incorrect instance handles for keyed data tables. This caused applications receiving the replayed data to treat multiple instances as a single instance, possibly leading to data loss or other unexpected behavior. This problem has been resolved.

[RTI Issue ID RECORD-794]

5.2.9 Converter Crashed when Deserializing Type that Yielded More Than 5000 Columns

The **-deserialize** option for *Converter* can only work with types that don't surpass the number-of-columns limit in the embedded SQLite® database, which is 5,000. *Converter* did not detect this case properly and crashed. This problem has been resolved.

[RTI Issue ID RECORD-795]

5.2.10 Recorder Crashed when Parsing XML Types: unresolved symbols

The following XML configuration (under the <file_group> tag used to configure types in the *Record* tool) caused a crash if there were undefined symbols (e.g., constant value not found) in the **TypesFileB.xml** file:

```
<element>
    <file_name>
        <element>TypesFileA.xml</element>
    </file_name>
    <type>
        <element>
            <type_name>TypeA</type_name>
            <topics>
                <element>TopicA</element>
            </topics>
        </element>
    </type>
</element>

<element>
    <file_name>
        <element>TypesFileB.xml</element>
    </file_name>
    <type>
        <element>
            <type_name>TypeB</type_name>
            <topics>
                <element>TopicB</element>
            </topics>
        </element>
    </type>
</element>
```

Note: This did not crash if the files were all specified under the same <element> entry. This problem has been resolved.

[RTI Issue ID RECORD-815]

5.2.11 Failure to Replay All Samples in Keyed Topic Table

The *Replay* tool did not correctly retrieve the number of samples in a keyed topic table that contained disposed or unregistered entries. This caused it to replay less samples than it should have. This problem has been resolved.

[RTI Issue ID RECORD-822]

5.2.12 RTI Replay fails with "serialization error: insufficient space" Error

The *Replay* tool triggers a DynamicData serialization error due to insufficient space when trying to reproduce a database containing mutable types. You would have seen these errors:

```
...
DL Error: :
DDS_DynamicDataTypePlugin_process_primitive_cdr_value:serialization error:
insufficient space
DL Error: : DDS_DynamicDataTypePlugin_cdr_to_parametrized_cdr:error copying
CDR value
DL Error: : DDS_DynamicDataTypePlugin_cdr_to_parametrized_cdr:error
converting from CDR to extended CDR
DL Error: : DDS_DynamicDataTypePlugin_cdr_to_parametrized_cdr:error
converting from CDR to extended CDR
DL Error: : DDS_DynamicDataTypePlugin_cdr_to_parametrized_cdr:error
converting from CDR to extended CDR
DL Error: : DDS_DynamicDataTypePlugin_serialize:error converting from CDR
to extended CDR
DL Error: : PRESWriterHistoryDriver_initializeSample:!serialize
DL Error: : WriterHistoryMemoryPlugin_addEntryToSessions:!initialize sample
DL Error: : WriterHistoryMemoryPlugin_getEntry:!add virtual sample to
sessions
DL Error: : WriterHistoryMemoryPlugin_addSample:!get entry
DL Error: : PRESWriterHistoryDriver_addWrite:!add_sample
DL Error: : PRESPsWriter_writeInternal:!collator addWrite
DL Debug: : PLAYBACKTopic_writeOneSerialized:Failed to write serialized
DynamicData sample
DL Debug: : PLAYBACKSession_run:!write serialized
DL Debug: : Terminating service...
DL Debug: : PLAYBACKSession_stop:!session:[map_horizons_session] 'stop'
command not valid in this state
DL Debug: : PLAYBACKDatabase_stop:!stop playback session
DL Debug: : PLAYBACKService_stop:!stop playback database
DL Debug: : Replay Service stopped
```

This problem happened because of two other bugs: RECORD-754, in which *Recording Service* wouldn't store the sample with the adequate format, and CORE-7744, in which the serialized sample size calculation for DynamicData samples with types that contained large numbers of mutable members was calculated incorrectly. Both of those bugs have been fixed, so this bug should no longer be seen.

[RTI Issue ID RECORD-823]

5.2.13 Replay Launched from Recording Console did not Contain All XML Type Information

When the *Replay* tool was launched from *Recording Console* with the configuration-by-file option, if the XML configuration file passed to the tool contained more than one element within the <type_config><xml><file_group> tag, the rest of the entries were ignored.

Consequently, if there was no type representation in the recorded DCPSPublication table for a topic, the tool did not have the corresponding XML type to use; thus the topic could not be replayed. This problem has been resolved.

[RTI Issue ID RECORD-826]

5.2.14 Replay not Launched Properly from Recording Console if using Patched Release

Recording Console did not properly launch the *Replay* tool after a patch was installed. This occurred when a *Recording Service* patch had been installed on top of an existing *Connext DDS* installation. The error happened because *Recording Console* was overriding environment variables when launching *Replay* with old values. This problem has been resolved.

[RTI Issue ID RECORD-829]

5.2.15 Replay Crashed with Missing Tag in XML

The *Replay* tool crashed if the <record_group_name> tag inside a <replay_topic><input> tag was missing. This tag is required. We have added code to enforce the presence of the tag; if it is missing parsing of the XML will fail.

[RTI Issue ID RECORD-832]

5.2.16 Replay Fails to Write Sample, Serialization Error

When using the *Replay* tool, you may have seen the following error for some types:

```
DL Debug: : RTI Replay started
DL Error: : PRESWriterHistoryDriver_initializeSample:!serialize
DL Error: : WriterHistoryMemoryPlugin_addEntryToSessions:!initialize sample
DL Error: : WriterHistoryMemoryPlugin_getEntry:!add virtual sample to ses-
sions
DL Error: : WriterHistoryMemoryPlugin_addSample:!get entry
DL Error: : PRESWriterHistoryDriver_addWrite:!add_sample
DL Error: : PRESPsWriter_writeInternal:!collator addWrite
DL Debug: : PLAYBACKTopic_writeOneSerialized:Failed to write serialized
DynamicData sample
DL Debug: : PLAYBACKSession_run:!write serialized
DL Debug: : Terminating service...
```

This problem has been resolved.

[RTI Issue ID RECORD-848]

5.2.17 Potential Segmentation Fault when Shutting Down and Still Receiving Discovery Events

While shutting down, the *Record* tool may have had a segmentation fault when new discovery events were processed. This happened because of a race condition between the instrumented discovery callback for the DCPSPublication reader and finalization of an internal event queue.

This problem has been resolved.

[RTI Issue ID RECORD-870]

---

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. Most SQLite database viewer tools include functionality to export the database contents to other formats such as XML or CSV. In cases where the database was recorded with filtered fields, it's possible to use one of these tools to export the data.

- 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.
- When using *Recording Service's* remote shell and configuring it remotely, if the new file's path is too long, it may fail and show the following error:

  ```
  RTI Record Shell> configure advanced_example -remotefile <very long path
  to XML file>
  Unknown command: [A]
  ```

  [RTI Issue ID RECORD-801]

- 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 DDS* 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]
- Assignment Data Transformation only Supports Assignment of Primitive Fields not Part of Arrays or Sequences

  The data transformation library distributed with *Routing Service* only supports the assignment of primitive fields (including strings) that are not part of arrays or sequences.

  For example:

  ```
  < transformation className="TestTransformationLib::FieldMapping">
    <expression></expression>
    <parameter>position.x=position.y</parameter> <!- This is supported ->
    <parameter>x=y</parameter> <!- This is supported ->
    <parameter>x[0]=y[0]</parameters> <!- This is not supported ->
    <parameter>position=position</parameter> <!- This is not supported ->
  < /transformation>
  ```

  For more details about data transformation, see Chapter 3 in the *Routing Service User's Manual*.