RTI Security Plugins

Installation Guide

Version 7.3.0

Trademarks

Copy and Use Restrictions

Third-Party Software

Notices

Deprecations and Removals

Any deprecations or removals noted in this document serve as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220 and/or any other agreements by and between RTI and customer regarding maintenance and support of RTI's software.

Deprecated means that the item is still supported in the release, but will be removed in a future release. *Removed* means that the item is discontinued or no longer supported. By specifying that an item is deprecated in a release, RTI hereby provides customer notice that RTI reserves the right after one year from the date of such release and, with or without further notice, to immediately terminate maintenance (including without limitation, providing updates and upgrades) for the item, and no longer support the item, in a future release.

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

Chapter 1 Download Instructions

> The *Connext* evaluation and LM bundles include the *Security Plugins* and OpenSSL®. If you have the evaluation or LM version of *Connext* (with "eval" or "lm" in its package name), you can skip to the next chapter.

Log into the RTI Customer Portal, https://support.rti.com/. You will need your username and password, which are included in the letter confirming your purchase. If you do not have this letter, please contact license@rti.com If you need help with the download process, contact support@rti.com.

Once you have logged into the portal, select the **Downloads** link. Which files to download depends on your host, target, and which crypto libraries you will be using (OpenSSL or wolfSSL®). See details below.

1.1 Downloading the Security Plugins and OpenSSL

From the portal's **Downloads** page, select the appropriate version of the *Security Plugins* and OpenSSL for your platform. (You may also obtain OpenSSL from another source.)

1. For the *Security Plugins*, download both:
   - **rti_security_plugins-7.3.0-host-openssl-3.0-<host platform>.rtipkg**
   - **rti_security_plugins-7.3.0-target-openssl-3.0-<target architecture>.rtipkg**

   The host package includes the compiler-independent *Security Plugins* dependencies (documentation, headers, and the libraries used by RTI tools and services) for the host platform.

   The target package contains the *Security Plugins* libraries that you will link your application against.

2. For OpenSSL, download both:
   - **openssl-3.0.12-7.3.0-host-<host platform>.rtipkg**
   - **openssl-3.0.12-7.3.0-target-<target architecture>.rtipkg**

   The host package includes the OpenSSL distribution files for RTI tools and services.

   The target package includes OpenSSL distribution files that you will link your application against.

   Note: The OpenSSL target packages for cross-compiled architectures such as QNX do not include the OpenSSL executable.

Where:

- **<host platform>** depends on your host (**x64Linux** for Linux systems, **darwin** for macOS systems, **x64Win64** for Windows systems).
- **<target architecture>** names are described in the *RTI Connext Core Libraries Platform Notes*.

1.2 Downloading the Security Plugins for use with wolfSSL

From the portal's **Downloads** page, select the appropriate version of the *Security Plugins* for wolfSSL. Only select target architectures can be used with wolfSSL; these are noted in the Compatibility section of the *Security Plugins Release Notes*.

1. For the *Security Plugins*, download both:
   - **rti_security_plugins-7.3.0-host-wolfssl-5.5 -<host platform>.rtipkg**
   - **rti_security_plugins-7.3.0-target-wolfssl-5.5-<target architecture>.rtipkg**

   The host package includes the compiler-independent *Security Plugins* dependencies (documentation, headers, and the libraries used by RTI tools and services) for the host platform.

   The target package contains the *Security Plugins* libraries that you will link your application against.
2. For wolfSSL:

   RTI does not distribute wolfSSL as a package bundle. You should get a commercial version of wolfSSL 5.5.1 and follow the instructions in 3.3 Installing a Crypto Library on page 7.

Chapter 2 Paths Mentioned in Documentation

The documentation refers to:

- **<NDDSHOME>**

  This refers to the installation directory for *RTI® Connext®*. The default installation paths are:

  - macOS® systems:
    /Applications/rti_connext_dds-7.3.0
  - Linux systems, non-*root* user:
    /home/<*your user name*>/rti_connext_dds-7.3.0
  - Linux systems, *root* user:
    /opt/rti_connext_dds-7.3.0
  - Windows® systems, user without Administrator privileges:
    <*your home directory*>\rti_connext_dds-7.3.0
  - Windows systems, user with Administrator privileges:
    C:\Program Files\rti_connext_dds-7.3.0

  You may also see **$NDDSHOME** or **%NDDSHOME%**, which refers to an environment variable set to the installation path.

  Wherever you see **<NDDSHOME>** used in a path, replace it with your installation path.

Note for Windows Users: When using a command prompt to enter a command that includes the path **C:\Program Files** (or any directory name that has a space), enclose the path in quotation marks. For example:

```
"C:\Program Files\rti_connext_dds-7.3.0\bin\rtiddsgen"
```

Or if you have defined the **NDDSHOME** environment variable:

```
"%NDDSHOME%\bin\rtiddsgen"
```

- ***<path to examples>***

By default, examples are copied into your home directory the first time you run *RTI Launcher* or any script in **<NDDSHOME>/bin**. This document refers to the location of the copied examples as ***<path to examples>***.

Wherever you see ***<path to examples>***, replace it with the appropriate path.

Default path to the examples:

- macOS systems: **/Users/<*your user name*>/rti_workspace/7.3.0/examples**
- Linux systems: **/home/<*your user name*>/rti_workspace/7.3.0/examples**
- Windows systems: ***<your Windows documents folder>*\rti_workspace\7.3.0\examples**

  Where 'your Windows documents folder' depends on your version of Windows. For example, on Windows 10, the folder is **C:\Users\<*your user name*>\Documents**.

Note: You can specify a different location for **rti_workspace**. You can also specify that you do not want the examples copied to the workspace. For details, see *Controlling Location for RTI Workspace and Copying of Examples* in the *RTI Connext Installation Guide*.

Chapter 3 Installation Instructions

You do not need administrator privileges. All directory locations are meant as examples only; adjust them to suit your site.

Follow the steps in either:

3.1 Installing an Evaluation or LM Version

> The evaluation ("eval") or LM ("lm") version of *Connext* comes with OpenSSL, which will be automatically installed. wolfSSL is not supported with the Evaluation or LM version.

1. Install the evaluation ("eval") or LM ("lm") bundle as described in the RTI Connext Installation Guide.

   The "eval" and "lm" bundles include *Security Plugins* and OpenSSL. The installer provides a pre-built version of OpenSSL 3.0.12. If you want to build your own version of OpenSSL 3.0.12, you can find the source code here: https://www.openssl.org/source/.

   After installation, the *Security Plugins* header files and libraries will be in **<install dir>/include/ndds/security** and **<install dir>/lib/<target architecture>**, respectively. OpenSSL will be under **<install dir>/third_party**.
2. Add OpenSSL's **/bin** directory to your PATH.

   For example, assuming you want to use the *release* version of the OpenSSL libraries, enter the following command (all on one line). Adjust the path to match your installation directory and use your own architecture string.

On Linux and macOS systems:

```
> export PATH=
  <install dir>/third_party/openssl-3.0.12/<architecture>/release/bin:${PATH}
```

If linking dynamically, also add OpenSSL's **/lib** directory in your **LD_LIBRARY_PATH**. For example:

```
> export LD_LIBRARY_PATH=
  <install dir>/third_party/openssl-3.0.12/<architecture>/release/lib:$LD_LIBRARY_PATH
```

On Windows systems:

```
> set PATH=
  <install dir>\third_party\openssl-3.0.12\<architecture>\release\bin;%PATH%
```

3. To verify your installation, enter:

```
> openssl version
```

You should see this response:

```
OpenSSL 3.0.12
```

4. Your *Security Plugins* distribution requires a license file. See Chapter 4 License Management on page 14.

3.2 Installing a Regular Version

Please do not install more than one *Security Plugins* host package per host architecture or more than one *Security Plugins* target package per target architecture. If you attempt to do so (for example, if you install both **rti_security_plugins-7.3.0-target-openssl-3.0-<target architecture>.rtipkg** and **rti_security_plugins-7.3.0-wolfssl-5.5-target-<target architecture>.rtipkg**), then the earliest package will be relegated to a backup directory (see Special Backup of RTI Libraries, in the RTI Connext Installation Guide)), and the latest package will overwrite any intermediate packages.

3.2.1 Security Plugins for OpenSSL

1. Install the *Connext* host and target bundles as described in the RTI Connext Installation Guide.
2. Install the *Security Plugins* host and target packages:
   - **rti_security_plugins-7.3.0-host-openssl-3.0-<host platform>.rtipkg**
   - **rti_security_plugins-7.3.0-target-openssl-3.0-<target architecture>.rtipkg**

   <host platform> depends on your host (such as **x64Linux** on Linux systems, **darwin** on macOS systems, **x64Win64** on Windows systems).

   <target architecture> is one of the supported platforms, see the *RTI Security Plugins Release Notes*.

After installation, the *Security Plugins* header files and libraries will be in **<install dir>/include/ndds/security** and **<install dir>/lib/<target architecture>**, respectively.

3. Install OpenSSL as described in 3.3.1 Installing OpenSSL below.

3.2.2 Security Plugins for wolfSSL

The *Security Plugins* for wolfSSL are only supported on select target platforms. The Compatibility section of the *RTI Security Plugins Release Notes* lists which platforms are compatible with wolfSSL.

1. Install the *Connext* host and target bundles as described in the *RTI Connext Installation Guide*.
2. Install the *Security Plugins* host and target packages that are compatible with wolfSSL:
   - **rti_security_plugins-7.3.0-wolfssl-5.5.1-host-<host platform>.rtipkg**
   - **rti_security_plugins-7.3.0-wolfssl-5.5.1-target-<*target architecture*>.rtipkg**

   <host platform> depends on your host (**x64Linux** on Linux systems, **darwin** on macOS systems, **x64Win64** on Windows systems).

   <target architecture> is one of the supported platforms that supports wolfSSL, see the *RTI Security Plugins Release Notes*.

   After installation, the *Security Plugins* header files and libraries will be in **<install dir>/include/ndds/security** and **<install dir>/lib/<target architecture>**, respectively.
3. Install wolfSSL as described in 3.3 Installing a Crypto Library below.

3.3 Installing a Crypto Library

3.3.1 Installing OpenSSL

> If you have the evaluation or LM version of *Connext* (with "eval" or "lm" in the package file name): OpenSSL 3.0.12 is installed automatically with the bundle. The following instructions are only for regular installations.

RTI provides:

- An OpenSSL host package, which enables OpenSSL for RTI's applications such as *RTI Admin Console*, *RTI Routing Service*, *rtiddsspy*, etc.
- An OpenSSL target package, which provides OpenSSL libraries that can be used to secure your applications.

3.3.1.1 Linux and macOS Systems

1. Make sure you've installed host and target *Security Plugins* packages as described in 3.2.1 Security Plugins for OpenSSL on page 6.
2. Install an OpenSSL host package from RTI:
   openssl-3.0.12-7.3.0-host-<host platform>.rtipkg.

   The **<host platform>** is **x64Linux** for Linux systems, or **darwin** for macOS systems. Use the same process that you used for the **.rtipkg** files in the previous step.

3. Install an OpenSSL target package from RTI:
   openssl-3.0.12-7.3.0-target-<target architecture>.rtipkg.

   Use the same process that you used for the **.rtipkg** files in the previous step.

4. **Only on cross-compiled architectures:** The OpenSSL target package for cross-compiled architectures does not include the openssl executable in the **bin** directory. If you require the OpenSSL executable (e.g., to generate identity certificates), you need to download and install the OpenSSL target package for an architecture that is not cross-compiled. Once the package is installed, add it to your PATH as described in the next step.
5. Include the resulting OpenSSL **bin** directory in your **PATH**. For example, assuming you want to use the "release" version of the OpenSSL 3.0.12 libraries (enter the command all on one line):

   ```
   export PATH=<NDDSHOME>/third_party/openssl-3.0.12/<architecture>/release/bin:${PATH}
   ```

6. If you will be using the dynamic libraries, include the resulting OpenSSL **lib** directory in your LD_LIBRARY_PATH (on Linux systems) or DYLD_LIBRARY_PATH (on macOS systems). For example, assuming you want to use the *release* version of the OpenSSL 3.0.12 libraries (enter the command all on one line):

   ```
   export LD_LIBRARY_PATH=<NDDSHOME>/third_party/openssl-
   3.0.12/<architecture>/release/lib:$LD_LIBRARY_PATH
   ```

7. To verify your OpenSSL installation, enter:

   ```
   openssl version
   ```

   You should see the version that you just installed:

   ```
   OpenSSL <version>
   ```

   If you see a version that you didn't expect, your PATH may be pointing with a higher precedence to a different version of OpenSSL. You may need to place the version you just installed first or earlier in your PATH.

Note: When running the **openssl version** command, you may run into this OpenSSL warning:

```
WARNING: can't open config file: [default openssl built-in path]/openssl.cnf
```

To resolve this issue, set the environment variable OPENSSL_CONF to the path to the **openssl.cnf** file you are using. For example (enter this all on one line):

```
export OPENSSL_CONF=
<NDDSHOME>/third_party/openssl-3.0.12/<architecture>/release/ssl/openssl.cnf
```

3.3.1.2 Windows Systems

1. Make sure you've installed host and target *Security Plugins* packages as described in 3.2.1 Security Plugins for OpenSSL on page 6.
2. Install an OpenSSL host package from RTI:
   openssl-3.0.12-7.3.0-host-x64Win64.rtipkg.

   Use the same process that you used for the **.rtipkg** files in the previous step.
3. Install an OpenSSL target package from RTI:
   openssl-3.0.12-7.3.0-target-<target architecture>.rtipkg.
4. Add the resulting OpenSSL **bin** directory to your **Path** environment variable. For example (enter the command all on one line):

   ```
   set PATH=
   <NDDSHOME>\third_party\openssl-3.0.12\<architecture>\release\bin;%PATH%
   ```
5. To verify your installation, enter:

   ```
   openssl version
   ```

   You should see the version that you just installed:

   ```
   OpenSSL <version>
   ```

   If you see a version that you didn't expect, your PATH may be pointing with a higher precedence to a different version of OpenSSL. You may need to place the version you just installed first or earlier in your PATH.

   Note: When running the **openssl version** command, you may run into this OpenSSL warning:

   ```
   WARNING: can't open config file: [default openssl built-in path]/openssl.cnf
   ```

   To resolve this issue, set the environment variable OPENSSL_CONF to the path to the **openssl.cnf** file you are using. For example (enter this all on one line):

   ```
   export OPENSSL_CONF=
   <NDDSHOME>/third_party/openssl-3.0.12/<architecture>/release/ssl/openssl.cnf
   ```

3.3.2 Building wolfSSL

wolfSSL is only for use with specific architectures noted in the Compatibility section of the RTI Security Plugins Release Notes.

RTI does not distribute wolfSSL. You should get a commercial version of wolfSSL. See the RTI Security Plugins Release Notes for compatible versions.

In a location of your choice, build wolfSSL for your target architecture. Read the chapter on "Building" in the wolfSSL User Manual.

You must build wolfSSL with the following flags:

- `--enable-smime`
- `--enable-opensslall`
- `--enable-opensslextra`
- `--enable-crl`
- `--enable-certgen`
- `--enable-des3`
- `--enable-reproducible-build`
- `--enable-certext`
- `--enable-aesgcm-stream`
- `-DWOLFSSL_PSS_SALT_LEN_DISCOVER`
- `--enable-static (if building statically)`
- `--enable-harden`

This flag is specific to non-ARM Linux systems:

- `--enable-aesni`

These flags are specific to QNX systems:

- `--enable-smallstack`
- `-DWOLFSSL_HAVE_MIN`
- `-DWOLFSSL_HAVE_MAX`

The flags that start with a double dash (--) are options that you must pass to the `./configure` command. You must also pass to this command the options that start with the **-D** prefix. These options must be part of the **CFLAGS** space-separated list. For example:

```
CFLAGS="-DWOLFSSL_PSS_SALT_LEN_DISCOVER"
```

Alternatively, you can export **CFLAGS** as an environment variable.

> Compiling wolfSSL using different flags is not supported. You must build wolfSSL with the flags presented in this section. Otherwise, your application may not work correctly, or it may even crash.

You will need the resulting installation directory when installing wolfSSL in the next section.

We refer to the wolfSSL installation directory as the folder created after building wolfSSL. This folder should contain **bin/**, **include/**, **lib/**, and **share/** directories. You can configure it when building wolfSSL by adding the **--prefix** and **--exec-prefix** flags during the **make install** step.

There is a linking issue that happens if the build machine doesn't have enough main memory. It can occur when you compile wolfSSL with debug information. You may see the following error message:

```
/bin/ld: final link failed: Memory exhausted
collect2: error: ld returned 1 exit status
```

Or this one instead:

```
/bin/ld: BFD version 2.23.52.0.1-16.el7 20130226 internal error, aborting at merge.c line
877 in _bfd_merged_section_offset
/bin/ld: Please report this bug.
collect2: error: ld returned 1 exit status
```

In both cases, try to build wolfSSL on another machine, update the version of **ld**, or modify the build flags to reduce the amount of memory required for the debug symbols.

A suggestion to reduce the memory footprint is to pass **C_EXTRA_FLAGS="-g1 -feliminate-unused-debug-symbols -fdebug-types-section"** to the `./configure` command. This flag will use the *minimal* debug information level, remove debug information for symbols that are not actually used, and attempt to be more efficient so that the linker can remove duplicates.

You can find more information on bugs #16139 and #13379 of the **binutils** project.

3.3.3 Installing wolfSSL

After you've built wolfSSL for your target architecture:

1. Make sure you've installed the host and target *Security Plugins* packages as described in 3.2 Installing a Regular Version on page 6.
2. In your **<NDDSHOME>/third_party** directory, create **wolfssl-5.5.1/<target architecture>/release/**. Copy your wolfSSL installation directory under the **release/** folder.

(This assumes that you want to use the *release* version of the wolfSSL libraries, if you want to use the *debug* version of the libraries, use **<NDDSHOME>/third_party/wolfssl-5.5.1/<target architecture>/debug/** instead.)

You will end up with:
<NDDSHOME>/third_party/wolfssl-5.5.1/<target architecture>/[release|debug]/.

3. (This step isn't necessary for a QNX target, because the tools and services are supported natively on QNX systems.)

   If your *target* architecture is on a Linux, macOS, or Windows system and you want to use RTI Tools and Infrastructure Services: you also need to build the wolfSSL library compiled for your *host* architecture. To do so, repeat the steps in 3.3 Installing a Crypto Library on page 7 and create a new wolfSSL installation directory with the library compiled for your *host* architecture.

   Once you have wolfSSL compiled for your host architecture, copy the dynamic library files (*.so) to the **<NDDSHOME>/resource/app/lib/<host architecture>/** directory. The dynamic library files are in the **lib/** directory of your wolfSSL installation directory.

   <host architecture> is one of these: **darwin, x64Linux, x64Win64**.

   You must copy both the release and debug versions, including symbolic links.

4. Include the wolfSSL **bin/** directory in your PATH.

   For example, assuming you want to use the "release" version of the wolfSSL libraries (enter the command all on one line):

   ```
   export PATH=<NDDSHOME>/third_party/wolfssl-5.5.1/<architecture>/release/bin:${PATH}
   ```

   If you will be using the dynamic libraries, include the wolfSSL **lib/** directory in your library search path (LD_LIBRARY_PATH on Linux systems, DYLD_LIBRARY_PATH on macOS systems, or Path on Windows systems). For example, assuming you want to use the release version of the wolfSSL libraries (enter the command all on one line):

   ```
   export LD_LIBRARY_PATH=
   <NDDSHOME>/third_party/wolfssl-5.5.1/<architecture>/release/lib:$LD_LIBRARY_PATH
   ```

5. To verify your installation, enter:

   ```
   wolfssl-config --version
   ```

   You should see this response:

   ```
   5.5.1
   ```

   If you get a version other than wolfSSL 5.5.1, your PATH may be pointing with a higher precedence to a different version of wolfSSL. You may need to place version 5.5.1 first or earlier in

your PATH.

Chapter 4 License Management

There's a distinction between a ***license file*** (or string) and a ***license***. When you buy an RTI product, such as *RTI Connext Professional*, you are ***licensed*** to use it.

Within a package, some components may have to be activated using a license file or string, while others may not. For example, when you install the *Security Plugins* package, the *Security Plugins* libraries (which you are licensed to use) do not require a license file or string.

Some components that do not require a license file in certain packages may require a license file in other packages. For example, the *Security Plugins* libraries require a license file in the evaluation ("eval") and LM ("lm") packages. If a component in your *Connext* distribution requires a license file, you will receive one from RTI.

This section describes how to manage a license file. If you have more than one license file from RTI, you can concatenate them into one file. A single license file can be used to run on any architecture and is not node-locked. You are not required to run a license server.

4.1 Installing the License File

Save the license file in any location of your choice; the locations checked by the plugin are listed below. You can also specify the location of your license file in *RTI Launcher*'s **Configuration** tab. Then *Launcher* can copy the license file to the installation directory or to the user workspace.

Each time your application starts, it will look for the license file in the following locations until it finds a valid license. (The properties are in the PropertyQosPolicy of the *DomainParticipant*.)

1. A property called **com.rti.serv.secure.license_string**. (Only if you have an evaluation ("eval") or "lm" version of *Connext*.)
2. A property called **dds.license.license_string**. (Only if you have an evaluation ("eval") or "lm" version of *Connext*.)

The above two **license_string** properties can be set to the content of a license file. (This may be necessary if a file system is not supported on your platform.) You can set the property either in source code or in an XML file.

If the content of the license file is in XML, special characters for XML need to be escaped in the license string. Special characters include: quotation marks (") (replace with "), apostrophes (') (replace with '), greater-than (>) (replace with >), less-than (<) (replace with <), and ampersands (&) (replace with &).

Example XML file:

```
<domain_participant_qos>
    <property>
        <value>
            <element>
                <name>dds.license.license_string</name>
                <value>contents of license file</value>
            </element>
        </value>
    </property>
</domain_participant_qos>
```

3. A property called **com.rti.serv.secure.license_file**. (Only if you have an evaluation ("eval") or "lm" version of *Connext*.)
4. A property called **dds.license.license_file**. (Only if you have an evaluation ("eval") or "lm" version of *Connext*.)

   The above two **license_file** properties can be set to the location (full path and filename) of a license file. (This may be necessary if a default license location is not feasible and environment variables are not supported.) You can set the property either in source code or in an XML file.

   Example XML to set **dds.license.license_file**:

```
<domain_participant_qos>
    <property>
        <value>
            <element>
                <name>dds.license.license_file</name>
                <value>path to license file</value>
            </element>
        </value>
    </property>
</domain_participant_qos>
```

5. In the location specified in the environment variable RTI_LICENSE_FILE, which you may set to point to the full path of the license file, including the filename.

   Note: When you run any of the scripts in the **<NDDSHOME>/bin** directory, this automatically sets the RTI_LICENSE_FILE environment variable (if it isn't already set) prior to calling the

executable. It looks for the license file in two places: your **rti_workspace** directory and the installation directory (NDDSHOME). (See Chapter 2 Paths Mentioned in Documentation on page 3.)

6. If you are running any of the tools/services as executables via **NDDSHOME/bin/<*executable script*>** or through *Launcher*:
   a. In your **rti_workspace/<*version*>** directory, in a file called **rti_license.dat**.
   b. In your **rti_workspace** directory, in a file called **rti_license.dat**.
   c. In **<NDDSHOME>** (the *Connext* installation directory), in a file called **rti_license.dat**.
7. If you are running your own application linked with *Connext* libraries:
   a. In your current working directory, in a file called **rti_license.dat**.
   b. In **<NDDSHOME>** (the *Connext* installation directory), in a file called **rti_license.dat**.

As *Connext* attempts to locate and read your license file, you may (depending on the terms of the license) see a message with details about your license.

If the license file cannot be found or the license has expired, your application may be unable to initialize, depending on the terms of the license. If that is the case, your application's call to **DomainParticipantFactory.create_participant()** will return null, preventing communication.

If you have any problems with your license file, please email support@rti.com.

4.2 Adding or Removing License Management

If your license file changes—for example, you receive a new license for a longer term than your original license—you do not need to reinstall.

However, if you switch from an evaluation ("eval") or LM ("lm") distribution of *Connext* to a regular distribution, or vice versa, RTI recommends that you first uninstall your original distribution before installing your new distribution. Doing so will prevent you from inadvertently using a mixture of libraries from multiple installations.

Chapter 5 Special Backup of RTI Libraries

The following information applies if you are installing a patch release (a release with four digits, such as 7.3.0.1) into an existing installation directory. It does not apply if you are installing a patch release into a new directory. See the instructions that come with your patch release to know whether it's okay to install into an existing directory or you should install into a new one. You should almost never install a three-digit release into an existing installation directory.

5.1 Overwriting Libraries

When installing a new RTI package that overwrites a given library in the **<NDDSHOME>/lib/<architecture>** directory, the installer will copy the library from the **<NDDSHOME>/lib/<architecture>** directory to the **<NDDSHOME>/lib/<architecture>/<current_installed_version>** directory. This copy will serve as a backup.

For example, if you install patch version 7.3.0.1 to the RTI core libraries for x64Win64VS2017, your overwritten core 7.3.0 libraries will be copied into the following directory before the 7.3.0.1 libraries are installed: **<NDDSHOME>/lib/x64Win64VS2017/7.3.0**.

If you install another patch later, before the 7.3.0.1 libraries are overwritten, then the 7.3.0.1 libraries will be copied into **<NDDSHOME>/lib/x64Win64VS2017/7.3.0.1**.

5.2 Installing Add-on Products when Using RTI Tools

RTI Tools commonly rely on the backup directory, so it is important to make sure its contents are correct, especially when you are using add-on products, such as *RTI Security Plugins*. To ensure the backup contents are correct, do the following:

Before installing a new patch (e.g., 7.3.0.1) that contains an add-on product, such as *Security Plugins*, always install the base version (e.g., **rti_security_plugins-7.3.0-host-<arch>.rtipkg**) of the add-on product, if you haven't already, before installing the patch release

version (e.g., **rti_security_plugins-7.3.0.1-host-<arch>.rtipkg**).

Installing the base version first ensures that when RTI Tools (such as *RTI Admin Console*, *RTI Monitor*, and *RTI System Designer*) are not part of the patch release (they usually are not), they keep using the base libraries they are built with instead of the patched libraries. You will receive an error otherwise.

(The RTI tool will look for the base libraries in the backup directory, such as **<NDDSHOME>/resource/app/lib/x64Win64VS2017/7.3.0**. But that backup directory either will not exist or will not contain all of the necessary base libraries if you didn't install the base version packages first. The tool will then look for the library in the patched directory. But if the patched library uses a function that is not implemented in the core library in the backup directory, an error occurs.)

Chapter 6 Next Steps

See the *RTI Security Plugins Getting Started Guide* and *User's Manual* for further information on setting up and using the *Security Plugins*. The *Getting Started Guide* introduces important concepts and includes hands-on exercises. In the *User's Manual*, make sure to read these chapters in particular:

- Libraries Required for Using RTI Security Plugins
- Restrictions when Using RTI Security Plugins

For descriptions and examples of the security configuration in this release, please consult the **hello_security** examples under the **rti_workspace/<*version*>/examples/connext_dds/[c, c++, java, cs]** directory.

The *Security Plugins* documentation and examples may be updated online between releases. Please see the RTI Community website (https://community.rti.com) for the most up-to-date documentation.