RTSDK C# 2.2.3.L1

INSTALLATION GUIDE

1 Overview

RTSDK packages are specific to the product language (C/C++, C#, or Java) and include both the Enterprise Transport API and Enterprise Message API products. This guide describes the procedure to install and build RTSDK CSharp, starting with RTSDK version 2.2.2.L1 or ETA/EMA 3.3.0.L1 and later.

The RTSDK supports open sourcing and uses standards-based, freely-available open source tools to provide additional flexibility and benefit.

Solution and project files target the .NET 8.0 by default. Prebuilt libraries are made available for NET6.0 and NET8.0 using Visual Studio 2022.

Note: RTSDK version 2.0.8.L1 or ETA 3.0.0.L1 is the initial package release for ETA C#. RTSDK version 2.1.3.L1 or EMA 3.1.0.L1 is the initial package release for EMA C#.

2 Requirements and Limitations

- The RTSDK CSharp package uses XUnit in its unit tests.
- The RTSDK CSharp library may be rebuilt using the solution file provided with RRG package.

Note: RTSDK CSharp build does require access to the Internet to download necessary external dependencies from NuGet when using GitHub. With CSharp RRG package, all dependencies are included in NuGetPackages.

Please check README in CSharp directory after obtaining the package (refer to Section 3) for specific versions and a complete list of dependencies.

Document Version: 2.2.3
Date of issue: December 2024
Document ID: RTSCSharp223IP.240

3 Obtaining RTSDK

You can obtain RTSDK CSharp in the following ways:

- Download RTSDK from LSEG. For details, see Section 3.1.
- Obtain RTSDK from GitHub. For details, see Section 3.2.
- Obtain RTSDK libraries from NuGet.org. For details, see Section 3.3.

If you download RTSDK from LSEG, please download two RRG packages: RTSDK RRG (language specific) and RTSDK BinaryPack RRG (contains libraries for all support platforms and languages). The BinaryPack RRG contains closed source libraries which permit users to build and link all dependent libraries to have a fully function product. To combine both RRG packages, see Section 3.1.1.

If you obtain RTSDK from GitHub, libraries and examples may be built with any required libraries including LSEG closed source libraries downloaded automatically from NuGet.org during build. For more information, see Section 3.2.2.

3.1 Download RTSDK Packages from LSEG

Download the RRG and BinaryPack RRG packages from the following locations:

- LSEG Software Downloads page: https://myaccount.lseg.com/content/mytr/en/downloadcenter.html.

  Search downloads for product family, "MDS - API", and product, "Real-Time SDK".
- Developer Community Portal: https://developers.lseg.com/en/api-catalog/real-time-opnsrc/rt-sdk-csharp/downloads

For the RRG package, the archive file names are in this format: **Real-Time-SDK-<version>.csharp.rrg**.

Starting with version 2.2.2.L1, the BinaryPack content is available as a separate RRG package, requiring you to download and extract two archives:

- RTSDK RRG package: **Real-Time-SDK-<***version***>.***csharp***.zip**.

  The extracted archive contains a **setup** directory with the package in this format: **RTSDK-<***version***>.CSharp.rrg.zip**. This package contains prebuilt libraries for supported .Net versions.

  Example: Download **Real-Time-SDK-2.2.2.L1.win.zip** which contains a setup directory containing the **RTSDK-2.2.2.L1.CSharp.rrg.zip** archive. The extracted **RTSDK-2.2.2.L1.CSharp.rrg** directory contains prebuilt libraries.
- RTSDK BinaryPack RRG package: **Real-Time-SDK-BinaryPack-<***version***>.zip**.

  The extracted archive contains a **setup** directory with the package in this format: **RTSDK-BinaryPack-<***version***>.rrg**. This package contains closed-source binaries. It is not platform specific and has content for all flavors/platforms of API. You can use this package to build RTSDK examples.

Note: To use the BinaryPack RRG package for building RTSDK examples, combine both packages as described in Section 3.1.1. This combined package contains all the necessary build files and external dependencies.

3.1.1 Using Binary Pack to Build RTSDK

To build RTSDK examples, combine the content of both packages before doing a build.

▶ **To combine content of the RRG and BinaryPack RRG:**

1. Download and extract both packages as described in Section 3.1.
2. Place the **RTSDK-BinaryPack-<***version***>.rrg** directory into the **RTSDK-<***version***>.CSharp.rrg** directory.

3. Rename **RTSDK-BinaryPack-<*version*>.rrg** to **RTSDK-BinaryPack**.

Example:

1. Download **Real-Time-SDK-2.2.2.L1.csharp.zip** and **Real-Time-SDK-BinaryPack-2.2.2.L1.zip**.
2. Extract content from the **setup** directory of each archive: **RTSDK-2.2.2.L1.CSharp.rrg** and **RTSDK-BinaryPack-2.2.2.L1.rrg**.
3. Move the **RTSDK-BinaryPack-2.2.2.L1.rrg** directory into the **RTSDK-2.2.2.L1.CSharp.rrg** directory.
4. Rename **RTSDK-BinaryPack-2.2.2.L1.rrg** to **RTSDK-BinaryPack**.

3.2 Obtain RTSDK from GitHub

To obtain RTSDK from GitHub, do one of the following:

- Download packages from GitHub
- Clone the GitHub repository

3.2.1 Download RTSDK Packages from GitHub

Download both source code and binary pack packages from the GitHub RTSDK releases page:

1. Browse to https://github.com/Refinitiv/Real-Time-SDK/releases.
2. From the **Assets** drop-down section, download the packages:
   - RTSDK RRG package: **Source code zip** or **tar.gz** archive.
   - RTSDK BinaryPack RRG package: **RTSDK-BinaryPack-<*version*>.zip** or **tar.xz** archive.

3.2.2 Clone GitHub Repository

Clone the RTSDK GitHub repository from https://github.com/Refinitiv/Real-Time-SDK.

To clone the repository, use the following command:

```
git clone https://github.com/Refinitiv/Real-Time-SDK.git
```

3.3 Obtain RTSDK Libraries from NuGet.org

You can specify RTSDK libraries are external dependencies downloadable from NuGet when building your application. Here are the dependencies to include in your **csproj** file:

```
<dependency>
  <ItemGroup>
    <PackageReference Include="LSEG.Eta.Core" Version="3.0.0" />
    <PackageReference Include="LSEG.Eta.ValueAdd" Version="3.0.0" />
    <PackageReference Include="LSEG.Eta.Ansi" Version="3.0.0" />
    <PackageReference Include="LSEG.Eta.AnsiPage" Version="3.0.0" />
    <PackageReference Include="LSEG.Ema.Core" Version="3.1.0"/>
  </ItemGroup>
</dependency
```

4 Building RTSDK

There are two ways to build the sources obtained from GitHub:

- Use the solution file to build libraries and examples. Refer to Build Using Solution Files and Visual Studio.
- Use `dotnet` command line to build the libraries and/or examples. Refer to Build Using dotnet Command Line.

Building RTSDK using `dotnet` command line is platform agnostic which means it works the same way on Linux and Windows platforms. Building using **Visual Studio** is applicable to Windows only.

The RRG package contains all required external dependencies in the **CSharp/NuGetPackages** directory. In an environment without Internet access, you must add this directory as a NuGet source and disable other NuGet sources for a build to succeed. Following are some `dotnet` commands to do so.

To check existing NuGet sources:

```
dotnet nuget list source
```

To add a new NuGet source:

```
dotnet nuget add source <full path to your RRG package/CSharp/NuGetPackages>
```

To disable certain NuGet sources:

```
dotnet nuget disable source <specify a source showed in the list>
```

Example:

```
dotnet nuget disable source "nuget.org"
```

4.1 Build Using Solution Files and Visual Studio

Use the provided solution (or **sln**) file to build in **Visual Studio**. Use the appropriate **Visual Studio** version.

4.2 Build Using dotnet Command Line

Navigate to **RTSDK/CSharp** and issue the appropriate `dotnet` command as follows to build libraries and/or examples:

```
dotnet build --configuration <Release|Debug> RTSDK.sln
```

Note:
- In a GitHub build, this builds libraries and places them into **Eta/Libs** or **Ema/Libs** and examples into **Eta/Executables** or **Ema/Executables**.
- In RRG package, this builds only libraries and places them into a custom directory: **Eta/Custom/Libs** or **Ema/Custom/Libs**.

By default, CSharp libraries and examples are built using **net8.x**. To build **.net6**, specify `-p:EsdkTargetFramework=net6.0` on the build command line. To build both **.net6** and **.net8**, specify `-p:EsdkTargetFramework=all` on the build command line.

For example:

```
dotnet build -p:EsdkTargetFramework=net8.0 -c Release RTSDK.sln
```

▶ To build just libraries

Sample command line to build libraries:

```
dotnet build --configuration Release Eta/Src/Core/Core.csproj
dotnet build --configuration Release Eta/Src/ValueAdd/ValueAdd.csproj
dotnet build --configuration Release Eta/Src/Ansi/Ansi.csproj
dotnet build --configuration Release Eta/Src/AnsiPage/AnsiPage.csproj
dotnet build --configuration Release Ema/Src/Core/EMA_Core.csproj
GitHub Only: dotnet build -t:Consumer --configuration Release RTSDK.sln
```

Note:
- In a GitHub build, this builds libraries and places them into **Eta/Libs** and examples into **Eta/Executables**
- In RRG package, this builds only libraries and places them into a custom directory: **Eta/Custom/Libs**

▶ To build just examples

Each example may be built separately using the individual **csproj** files. Please note that the RRG package also contains a **.sln** file for each example.

Sample command line to build examples:

```
dotnet build --configuration Release Eta/Applications/Consumer/Consumer.csproj
dotnet build --configuration Release Eta/Applications/Consumer/Consumer.sln
dotnet build --configuration Release Ema/Examples/Training/Consumer/100_Series/
   100_MP_Streaming/Cons100.csproj
```

Note:
- Both **sln** and **csproj** files build examples and place them into **Eta/Executables** or **Ema/Executables**.
- Example solution files only exist in the RRG package.
- In RRG package, building examples via **csproj** or **sln** link to pre-built libraries located in **Eta/Libs** or **Ema/Libs**.
- In a GitHub build, each example expects libraries in **Eta/Libs** or **Ema/Libs** to exist.

4.3 Running Examples

Navigate to the **CSharp** directory in the RTSDK package and issue the appropriate `dotnet` command to run various examples using

`dotnet` [runtime-options] [path-to-application] [arguments]

- `dotnet` Eta/Executables/Consumer/Debug/<.net version>/Consumer.dll [arguments]
- `dotnet` Eta/Executables/ConsMod1a/Debug/<.net version>/ConsMod1a.dll [arguments]
- `dotnet` Ema/Executables/Cons100/Debug/<.net version>/Cons100.dll [arguments]

```
dotnet Eta/Applications/VAConsumer/bin/Debug/net8.0/VAConsumer.dll -c localhost:14002
   DIRECT_FEED mp:TRI
```

Tip: You can see a list of all possible arguments by passing the command: `"-?"`

5 Package Directory Changes

The following figure illustrates the RTSDK package directory structure.

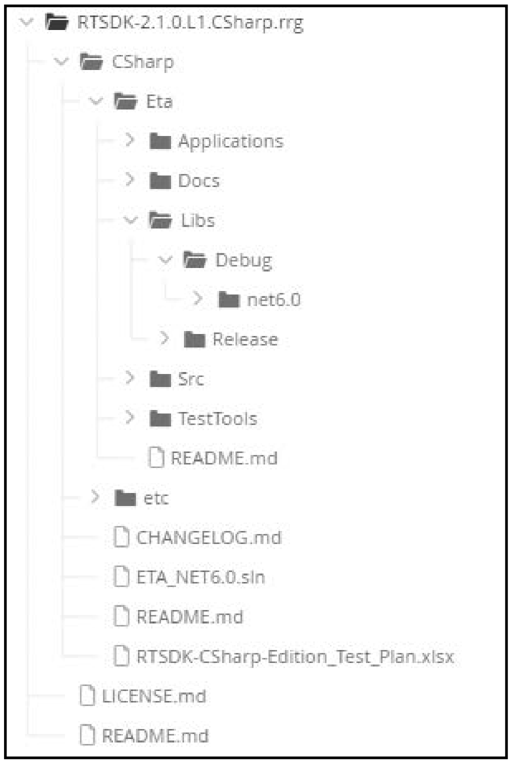
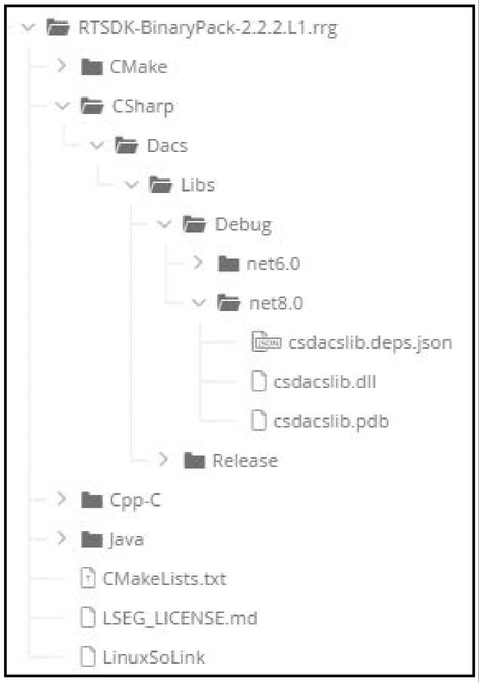
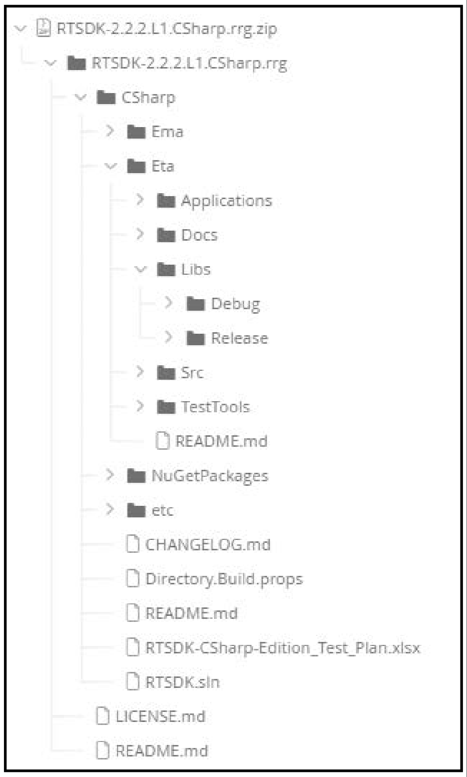
STRUCTURE STARTING WITH RTSDK 1.1.5	STRUCTURE STARTING WITH RTSDK 2.2.2	
RRG package:	BinaryPack RRG package:	RRG package:
 RTSDK-2.1.0.L1.CSharp.rrg CSharp Eta Applications Docs Libs Debug net6.0 Release Src TestTools README.md etc CHANGELOG.md ETA_NET6.0.sln README.md RTSDK-CSharp-Edition_Test_Plan.xlsx LICENSE.md README.md	 RTSDK-BinaryPack-2.2.2.L1.rrg CMake CSharp Dacs Libs Debug net6.0 net8.0 csdacslib.deps.json csdacslib.dll csdacslib.pdb Release Cpp-C Java CMakeLists.txt LSEG_LICENSE.md LinuxSoLink	 RTSDK-2.2.2.L1.CSharp.rrg.zip RTSDK-2.2.2.L1.CSharp.rrg CSharp Ema Eta Applications Docs Libs Debug Release Src TestTools README.md NuGetPackages etc CHANGELOG.md Directory.Build.props README.md RTSDK-CSharp-Edition_Test_Plan.xlsx RTSDK.sln LICENSE.md README.md

Table 1: RTSDK CSharp Package Structures

© LSEG 2023, 2024. All rights reserved.

Republication or redistribution of LSEG Data & Analytics content, including by framing or similar means, is prohibited without the prior written consent of LSEG Data & Analytics. 'LSEG Data & Analytics' and the LSEG Data & Analytics logo are registered trademarks and trademarks of LSEG Data & Analytics.

Any third party names or marks are the trademarks or registered trademarks of the relevant third party.

Document ID: RTSCSharp223IP.240
Date of issue: December 2024