RTSDK C# 2.2.0.L1

INSTALLATION GUIDE

1 Overview

RTSDK packages are specific to the product language (C/C++, C#, or Java). This guide describes the procedure to install and build RTSDK CSharp, starting with RTSDK version 2.0.8.L1 or ETA 3.0.0.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 6.0 platform and 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 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.

REFINITIV™

3 Obtaining the Package

You have the following options in obtaining the RTSDK:

- You can download the package from the Developer Community Portal at the following URL: https://developers.refinitiv.com/en/api-catalog/refinitiv-real-time-opnsrc/refinitiv-real-time-csharp-sdk/downloads

  You can also obtain the package from MyRefinitiv software downloads page as follows:

  a. Go to https://my.refinitiv.com/content/mytr/en/downloadcenter.html.

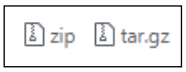
  b. Search for "MDS -API" and "Real-Time SDK" to locate the C# package.

  Note: RTSDK C# package downloaded from Developer Community Portal or MyRefinitiv software downloads contains the necessary build files and external dependencies.

- You can clone the RTSDK from the GitHub repository (at https://github.com/Refinitiv/Real-Time-SDK) by using the following command:

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

  Tip: You can also download the source from GitHub via the browser:

  - Browse to the URL https://github.com/Refinitiv/Real-Time-SDK/releases
  - Each release will have the following options listed beneath it's release name: 

  - To download a compressed package, click **zip** or **tar.gzip**.

- 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 Package Directory Changes

The following figure illustrates the RTSDK package directory structure.

RTSDK C# 2.2.0.L1 PACKAGE

The following figure illustrates the top-level directory structure for the RTSDK C# 2.2.0.L1 release:

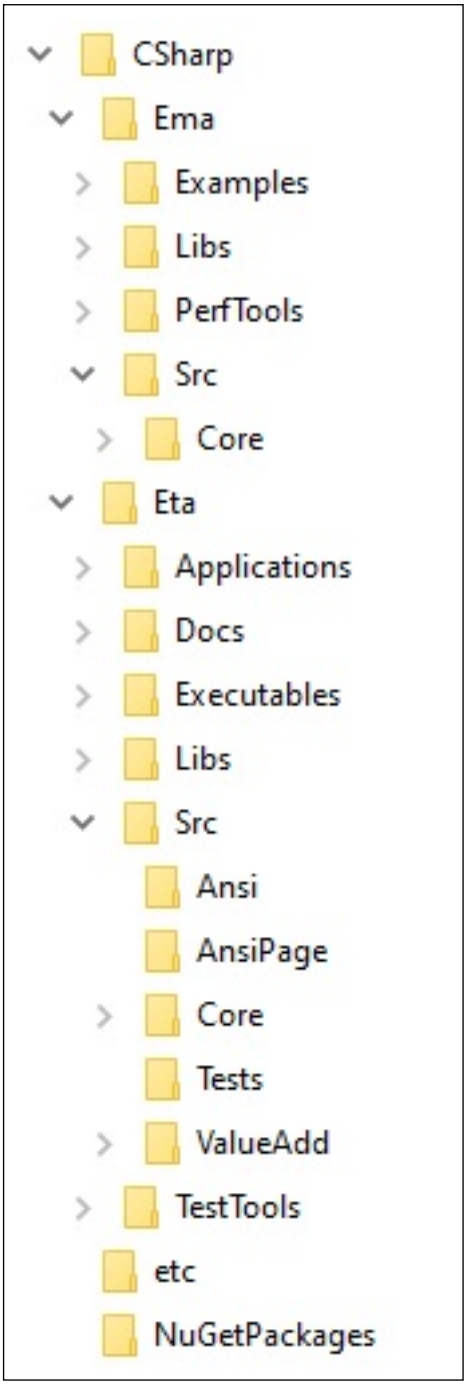


Figure 1. RTSDK CSharp Package Structure

5 Using the Package

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

- Use the solution file to build libraries and examples: Use appropriate **Visual Studio** version.
- Use **`dotnet`** command line to build the libraries and/or examples.

5.1 ▶ Building RTSDK

Building RTSDK using **`dotnet`** command lines 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"
```

Using Solution Files and Visual Studio

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

Using `dotnet`

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

```
dotnet build --configuration <Release|Debug> RTSDK_NET6.0.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.**

To build just libraries:

```
dotnet build --configuration Release Eta/Src/Core/Core_NET6.0.csproj
dotnet build --configuration Release Eta/Src/ValueAdd/ValueAdd_NET6.0.csproj
dotnet build --configuration Release Eta/Src/Ansi/Ansi_NET6.0.csproj
dotnet build --configuration Release Eta/Src/AnsiPage/AnsiPage_NET6.0.csproj
dotnet build --configuration Release Ema/Src/Core/EMA_Core_NET6.0.csproj
GitHub Only: dotnet build -t:Consumer --configuration Release RTSDK_NET6.0.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_NET6.0.csproj
dotnet build --configuration Release Eta/Applications/Consumer/Consumer_NET6.0.sln
dotnet build --configuration Release Ema/Examples/Training/Consumer/100_Series/
   100_MP_Streaming/Cons100_NET6.0.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.

▶ 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/net6.0/Consumer.dll [arguments]
- **`dotnet`** Eta/Executables/ConsMod1a/Debug/net6.0/ConsMod1a.dll [arguments]
- **`dotnet`** Ema/Executables/Cons100/Debug/net6.0/Cons100.dll [arguments]

```
dotnet Eta/Applications/VAConsumer/bin/Debug/net6.0/VAConsumer.dll -c localhost:14002
   DIRECT_FEED mp:TRI
```

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

RTSDK C# v2.2.0.L1 Installation Guide
Document Version: 2.2.0
RTSCSharp220IP.240