

RTSDK C# 3.0.0.L1

INSTALLATION GUIDE

1 Overview

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

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 CSharp 3.0.0.L1 is the initial package release for the C# language.

2 Requirements and Limitations

- The RTSDK CSharp package uses XUnit in its unit tests.
- The RTSDK CSharp library may be built 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.

2.1 External Dependencies

- K4os.Compression.LZ4
- Microsoft.IdentityModel.Tokens
- System.IdentityModel.Tokens.Jwt
- XUnit (for unit testing)

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

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>

Note: RTSDK CSharp package downloaded from Developer Community Portal contains the necessary build files. Upon build, all external dependencies are expected to be downloaded.

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

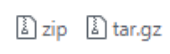
```
git clone https://github.com/Refinitiv/Real-Time-SDK.git
```



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.gz**.

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

```
<dependency>
<ItemGroup>
<PackageReference Include="LSEG.Eta.Ansi" Version="3.0.0" />
</ItemGroup>
</dependency>

<dependency>
<ItemGroup>
<PackageReference Include="LSEG.Eta.Core" Version="3.0.0" />
</ItemGroup>
</dependency>

<dependency>
<ItemGroup>
<PackageReference Include="LSEG.Eta.AnsiPage" Version="3.0.0" />
</ItemGroup>
</dependency>
```

```
<dependency>  
<ItemGroup>  
<PackageReference Include="LSEG.Eta.ValueAdd" Version="3.0.0" />  
</ItemGroup>  
</dependency>
```

4 Package Directory Changes

The following table illustrates the RTSDK package directory structure.

RTSDK C# 3.0.0.L1 PACKAGE
<p>The following diagram illustrates the top-level directory structure for the RTSDK C# 3.0.0.L1 release:</p> <pre>graph TD; CSharp[CSharp] --> Eta[Eta]; Eta --> Applications[Applications]; Eta --> Docs[Docs]; Eta --> Executables[Executables]; Eta --> Libs[Libs]; Eta --> Src[Src]; Eta --> Testtools[Testtools]; Eta --> etc[etc]; Applications --> AnsiPage[AnsiPage]; Applications --> Consumer[Consumer]; Applications --> ConsumerTest[ConsumerTest]; Applications --> EtaAppCommon[EtaAppCommon]; Applications --> NIPProvider[NIPProvider]; Applications --> PerfTools[PerfTools]; Applications --> Provider[Provider]; Applications --> Training[Training]; Applications --> VACommon[VACommon]; Applications --> VAConsumer[VAConsumer]; Applications --> VANIProvider[VANIProvider]; Applications --> VAProvider[VAProvider]; Src --> Ansi[Ansi]; Src --> AnsiPage[AnsiPage]; Src --> Common[Common]; Src --> Core[Core]; Src --> Tests[Tests]; Src --> Transport[Transport]; Src --> ValueAdd[ValueAdd]; Src --> QATools[QATools]; Src --> TransportTest[TransportTest];</pre>

Table 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.

► Building Libraries and Examples

1. Use the provided solution (or **sln**) file to build in Visual Studio.
2. Use **dotnet** to navigate to **RTSDK/CSharp**.
3. To build all: This command line builds both libraries and examples

```
dotnet build --configuration <Release|Debug> ETA_NET6.0.sln
```

4. 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
```

5. To build just examples: Each example may be built separately using the individual **csproj** files.
Sample Command Line:

```
dotnet build --configuration Release Eta/Applications/Consumer/Consumer_NET6.0.csproj
```

Note: Building the RTSDK on Linux is the same as building on Windows with the **dotnet** tool, however Visual Studio is only for Windows.

► 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]

© 2023 Refinitiv. All rights reserved.

Republication or redistribution of Refinitiv content, including by framing or similar means, is prohibited without the prior written consent of Refinitiv. 'Refinitiv' and the Refinitiv logo are registered trademarks and trademarks of Refinitiv and its affiliated companies.

RTSDK C# v3.0.0.L1 Installation Guide
Document Version: 3.0.0
RTSCSharp300IP.230



- **dotnet** Eta/Executables/Consumer/Debug/net6.0/Consumer.dll [arguments]
- **dotnet** Eta/Executables/ConsMod1a/Debug/net6.0/ConsMod1a.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: "-?"
