Contents

[Introduction 2](#_Toc102392268)

[Prerequisites 2](#_Toc102392269)

[Nur API 2](#_Toc102392270)

[Create the RFID sample project 2](#_Toc102392271)

[Adding NurApi to the project 2](#_Toc102392272)

[NurApiDotNet NuGet 3](#_Toc102392273)

[Create and install the application 3](#_Toc102392274)

[Application functionality 4](#_Toc102392275)

[Connecting to NUR device 4](#_Toc102392276)

[Reading RFID tags 4](#_Toc102392277)

# Introduction

This document describes how to create the FR22 RFID Sample application in C#. The source code for the sample can be found at [https://github.com/NordicID/fr22\_samples/tree/master/4. RfidSample](https://github.com/NordicID/fr22_samples/tree/master/4.%20RfidSample).

The RFID Sample application creates an application that connects to the RFID module via the NurApiDotnet library along with a simple Web UI for controlling it. Via the Web UI, the user can connect/disconnect to/from the RFID reader, start/stop the RFID tag inventory operation and view information about the visible RFID tags.

## Prerequisites

* FR22 device with OS version 0.8.0 or later
* fr\_appsigntool files from fr22\_appsigntool directory in <https://github.com/NordicID/fr22_samples/> (requires .NET Core 3.1)
* Visual Studio 2019/2022 with C# support installed
  + It’s possible to use VS Code or the dotnet tool, but this guide assumes Visual Studio is used.
* Knowledge of how the Hello World sample described in 1. FR22 Hello World.docx works
* Knowledge of how the RPC Demo sample described in 2. FR22 RPC demo.docx works
* Knowledge of how the RPC WebUI sample described in 3. FR22 RPC WebUI.docx works

## Nur API

NurApi is Nordic ID’s property protocol used to communicate with NUR (RFID) modules. NurApi is also the name of libraries that implement the NurApi protocol; there is a C library named NurApi and a C# API library named NurApiDotNet available as NuGet packages on nuget.org. The C# NurApiDotNet library uses the C NurApi library which in turn implements the NurApi protocol. The C NurApi library is available on the device (built into the firmware), but it can also be installed as part of the application zip. In this way the firmware version can be overridden in the application. This sample will use the NurApi library already available in the device firmware.

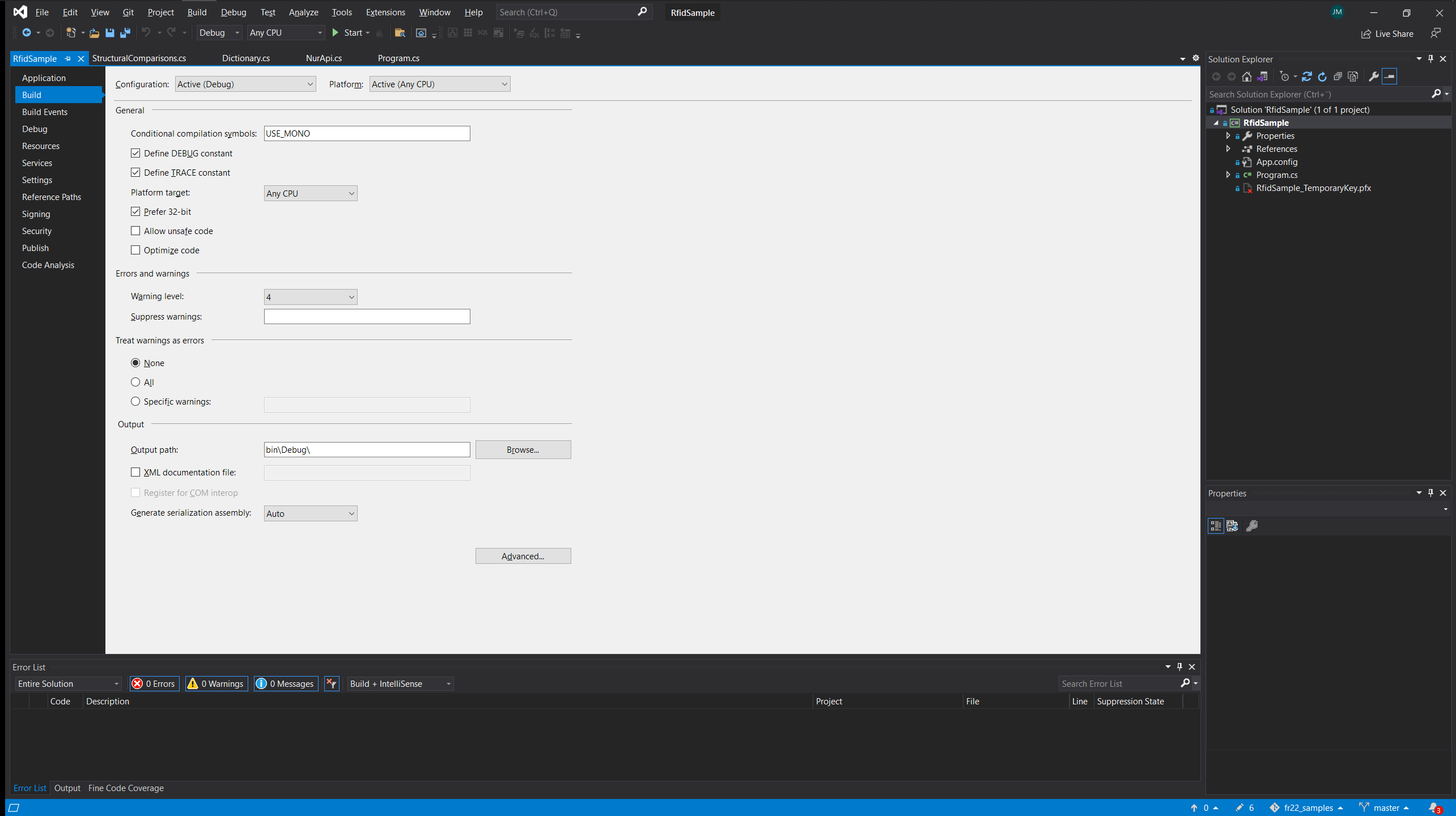
# Create the RFID sample project

Create a new RFID sample project similarly to how the RPC WebUI sample was created.

# Adding NurApi to the project

NurApiDotNet includes some platform dependant functionality. When making an application for the FR22 platform, the conditional compilation symbol *USE\_MONO* must be defined before adding the NurApiDotNet NuGet to the project (as when the NurApiDotNet NuGet is installed, it installs a different version of the library if the symbol is present).

In Visual Studio, this is done by right clicking the project name (*RfidSample*) and selecting *Properties*. From here, select the *Build* tab. Under the *General* section there is a *Conditional compilation symbols* to which the symbol *USE\_MONO* should be added:



## NurApiDotNet NuGet

The NordicID NurApiDotNet C# library is available as a NuGet package on nuget.org at <https://www.nuget.org/packages/NurApiDotNet/>. Install it to the project by:

* Open the NuGet package manager (in the Solution explorer, right click on either the project name or *References* and select *Manage NuGet Packages*)
* Select the browse tab and search for nurapidotnet (making sure nuget.org is the selected package source)

A screenshot of a computer

Description automatically generated with medium confidence

* Select the NurApiDotnet entry and press *Install* to begin the installation

# Create and install the application

The application is built, created and installed in the same way as the previous samples were.

After installation and application start, the application’s WebUI is available by navigating to *RfidSample* under the *Application* menu entry.

## Application functionality

The application automatically connects to the device’s internal NUR module (the NUR protocol uses TCP, so the sample could easily be extended to also connect to other devices) upon start-up. When connected, some basic information about the NUR module is shown. In addition, the sample WebUI provides buttons for starting/stopping RFID tag reading (inventory) and a list of the tags that have been read along with some basic tag data.

## Connecting to NUR device

The connection to the NUR module on localhost (127.0.0.1) is initialized with

\_nur.ConnectSocket("127.0.0.1", 4333);

(Note that on errors a NurApiException will be thrown). The NUR connection is available on port 4333 by default so this is where the sample connects to (note that the port can be changed in the RFID connection settings in the device WebUI).

## Reading RFID tags

NurApi provides several ways to read RFID tags by doing RFID inventories. This sample uses the stream-based inventory; first an inventory stream event listener is registered by calling:

\_nur.InventoryStreamEvent += new EventHandler<NurApi.InventoryStreamEventArgs>(OnInventoryStreamEvent);

After that inventory streaming is started by calling

\_nur.StartInventoryStream();

Calling StartInventoryStream without parameters will use the settings defined in the RFID settings of the device. These settings can be changed in the WebUI by navigating to *RFID/Settings*. NurApiDotNet also provides APIs for overriding the default settings set on the device.

Detailed documentation for NurApiDotNet is available by navigating to the its’ definitions in Visual Studio. The documentation is also available as a Microsoft Compiled HTLM (CHM) file in the Nur API SDK git at [https://github.com/NordicID/nur\_sdk/blob/master/dotnet/docs/NurApi .NET Documentation.chm](https://github.com/NordicID/nur_sdk/blob/master/dotnet/docs/NurApi%20.NET%20Documentation.chm). More NurAPI samples and documentation are available on Nordic ID’s public github at <https://github.com/NordicID>.