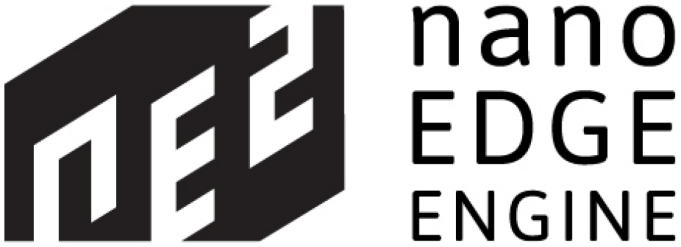
iSMA CONTROLLI

SDK

User Manual

Quick Start-up



iSMA CONTROLLI S.p.A. - Via Carlo Levi 52, 16010 Sant'Olcese (GE) - Italy | support@ismacontrolli.com

Table of Contents

1 Introduction

The nano EDGE ENGINE SDK is a software development toolkit allowing to develop new functionalities for device and application management. It contains a set of basic definitions required to write services, components, and extensions that can be plugged into a device workspace tree and become part of a user project.

1.1 Revision History

Date	Rev.	Description
6 Jun 2025	1.0	First edition

Table 1. Revision history

2 Step 1: Set Up Working Environment

The first step to start using the SDK is to set up a working environment. The nano EDGE ENGINE SDK is designed to work with the Microsoft's tool, Visual Studio.

- Download and install the Visual Studio tool: Visual Studio from Microsoft.

Note: Please pay attention to license conditions and select the appropriate license.

Note: The SDK is designed to work with the Visual Studio from the 2022 version.

Note: Please make sure that during Visual Studio installation an option for .NET desktop development is selected.

Install the .NET nanoFramework extension to the Visual Studio:

- Method 1:
  - in the Visual Studio, navigate to Extensions -> Manage Extensions;
  - on the left menu, choose 'Online';
  - type "nanoFramework" in a search bar;
  - install.
- Method 2:
  - download it from: .NET nanoFramework;
  - close all windows of Visual Studio and double click the downloaded .vsix file to install it;
  - upon complete installation, reopen the Visual Studio.

3 Step 2: Create Project

The next step is to create a new project in the Visual Studio:

- Firstly, log in to https://sdk.isma.lan and change the password to GitLab (otherwise, there might occur a problem with cloning the SDK repository from GitLab).
- Download the SDK source code: https://sdk.isma.lan/framework.sdk/framework.sdk.git (git clone https://sdk.isma.lan/framework.sdk/framework.sdk.git for Git Bash client).
- Run the Visual Studio.
- Choose a "Create a new project" option.

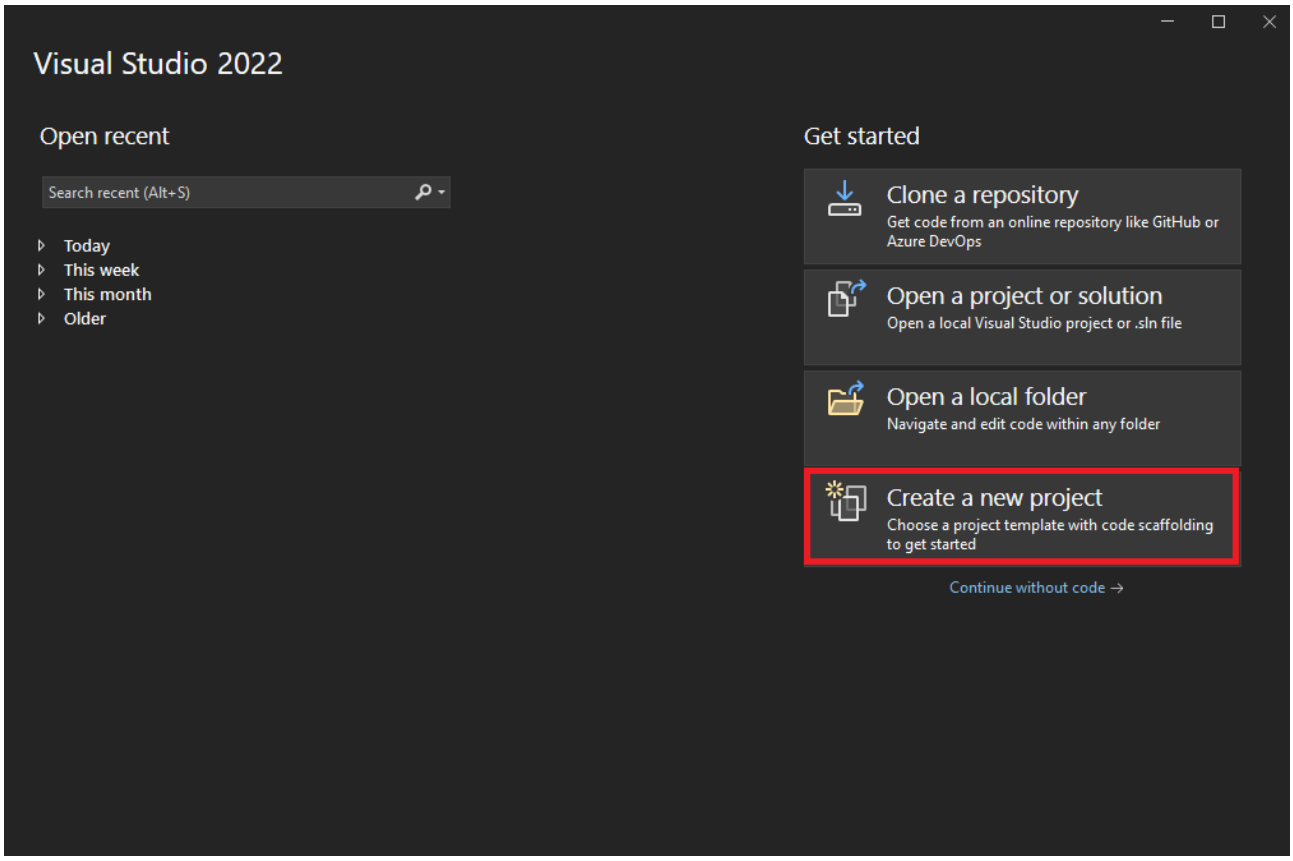


Figure 1. Create new project

- Type nanoFramework in a search box and choose "Class library (.NET nanoFramework). Click "Next".

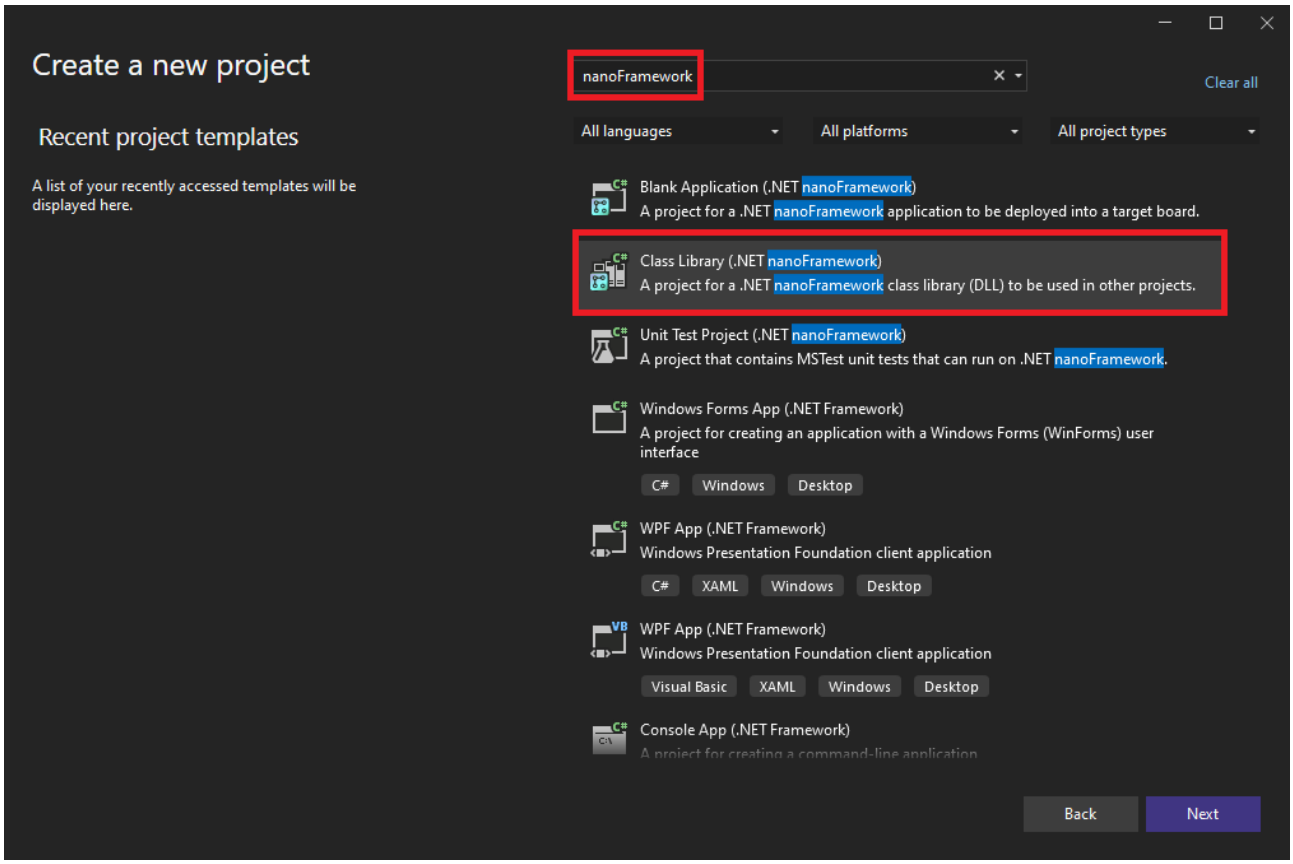


Figure 2. Search for nanoFramework extension

- Configure a new project's basic properties:
  - In the Project Name bar, type a name of the project. Make sure to set the project name according to a pattern: 'nameOfCompany_Math", for example, in case of iSMA CONTROLLI: 'iSMA_Math'.
  - In the Location bar, select a desired location.
  - Leave the "Place solution and project in the same directory" unchekced. Click "Create".

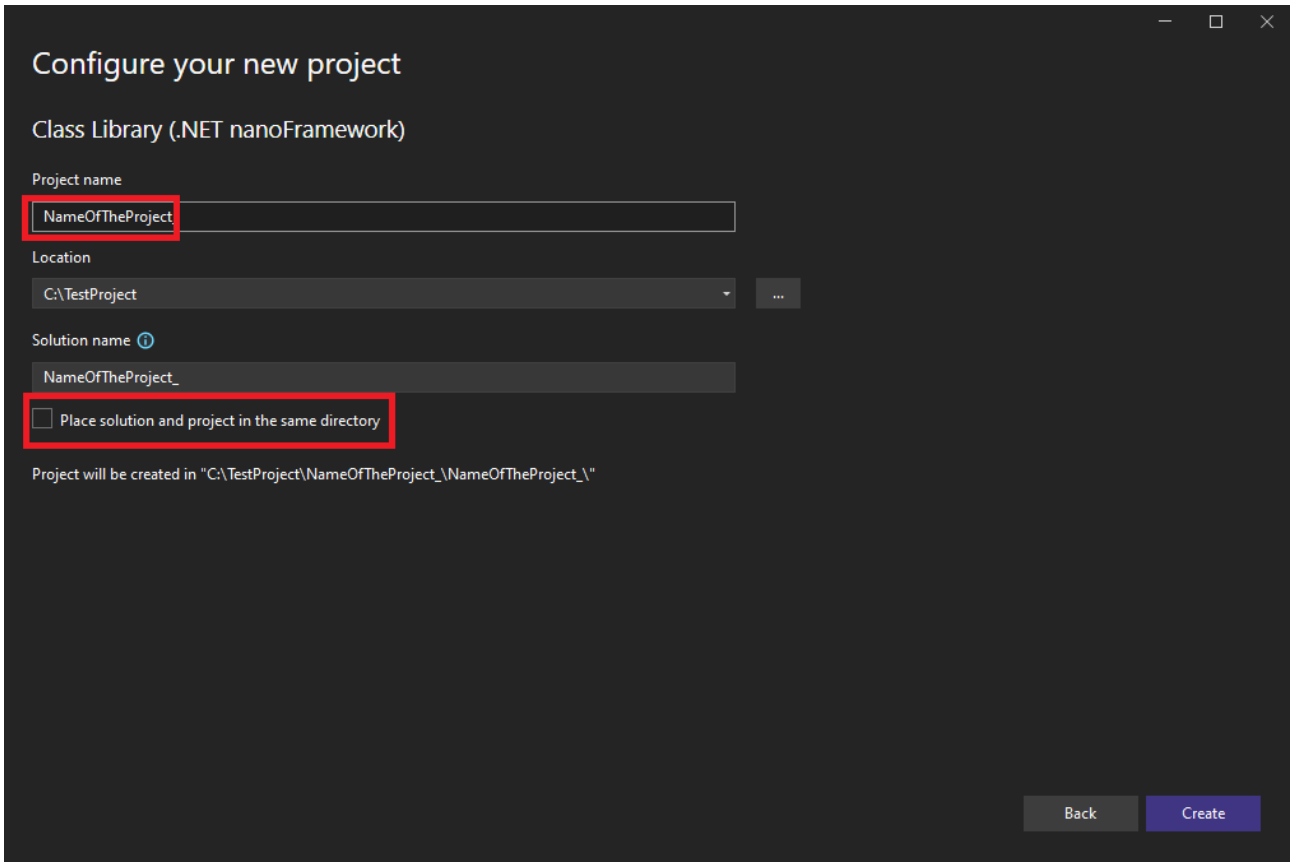


Figure 3. Configure basic properties of the project

The new project will be started and it will consist of a solution with single project attached. The project will only have one class called “Class1.cs”

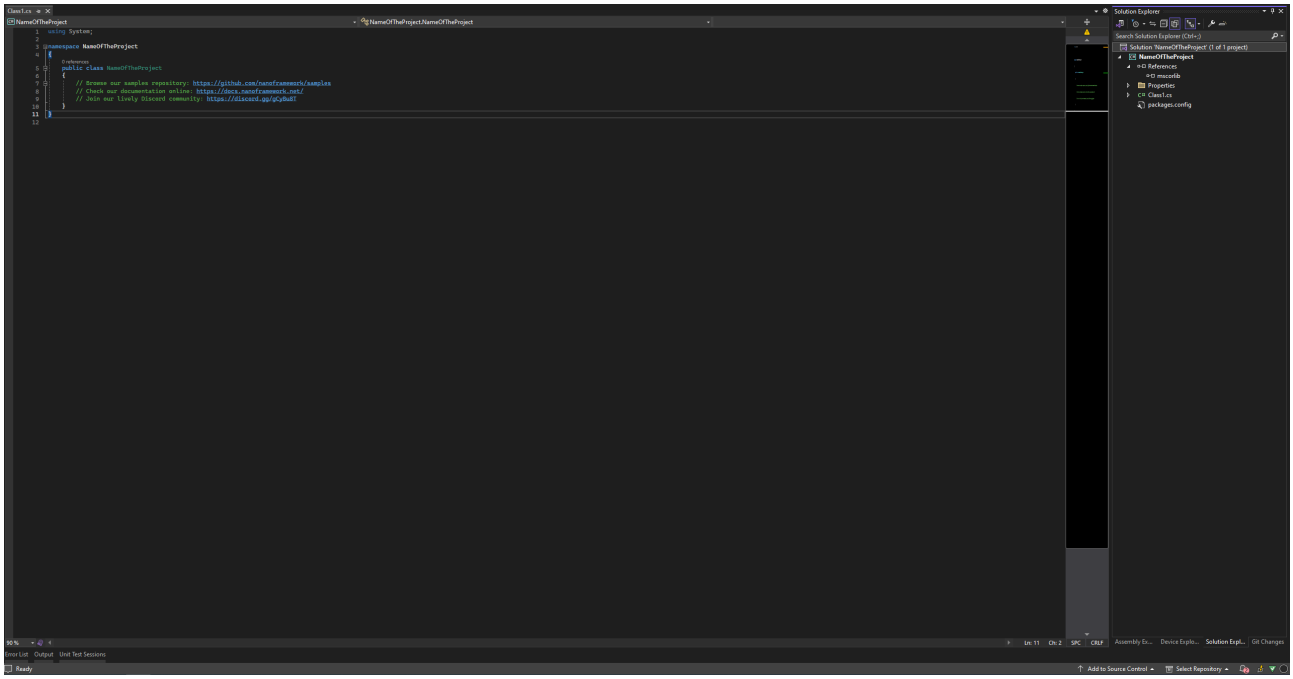


Figure 4. New project view

Now, it is required to reference the SDK source code:

- Right click on Solution, choose Add and Existing Project.

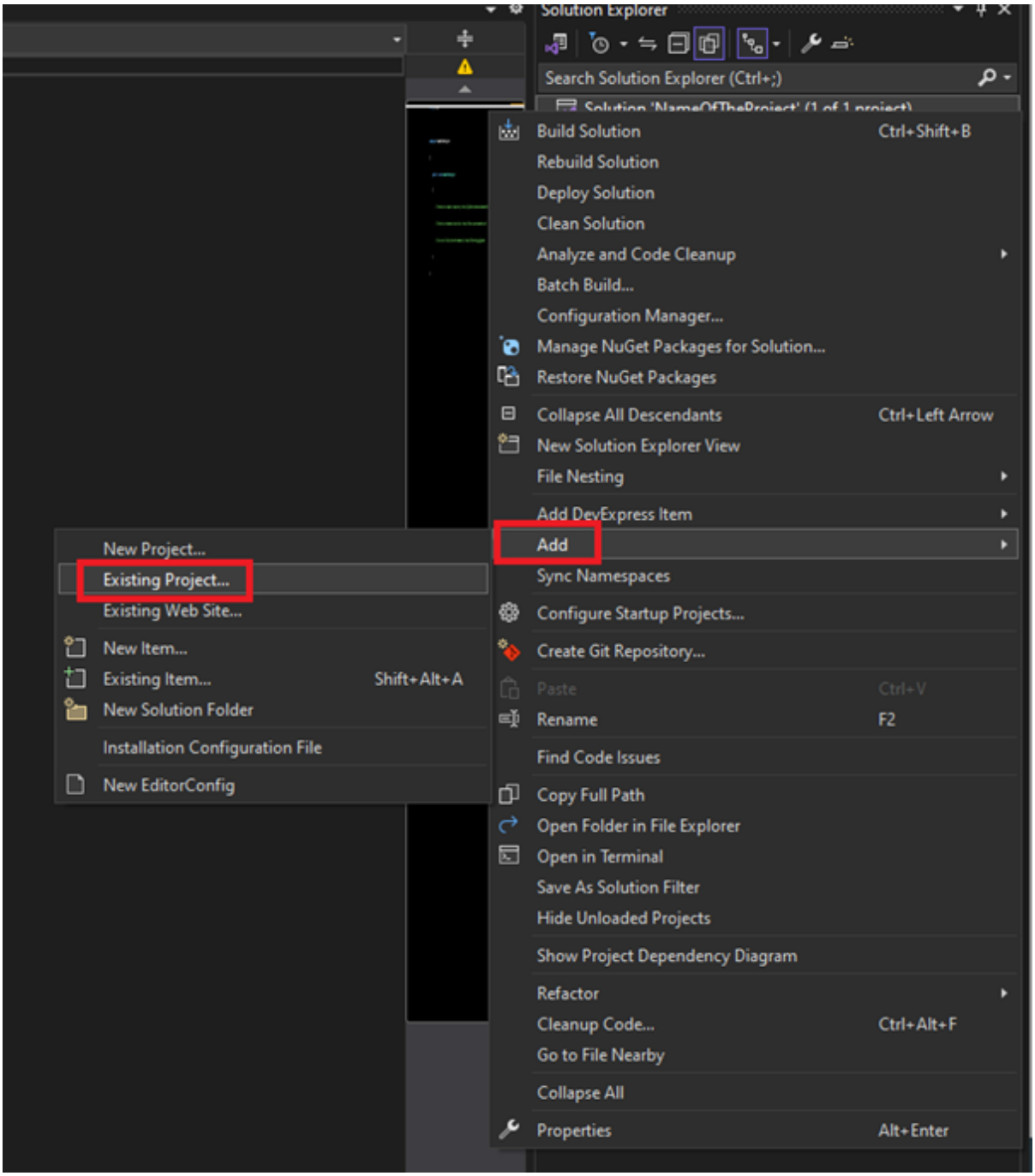


Figure 5. Adding SDK source code

- Navigate to the folder containing the source code of SDK and choose Framework.SDK.nfproj file. Click Open.

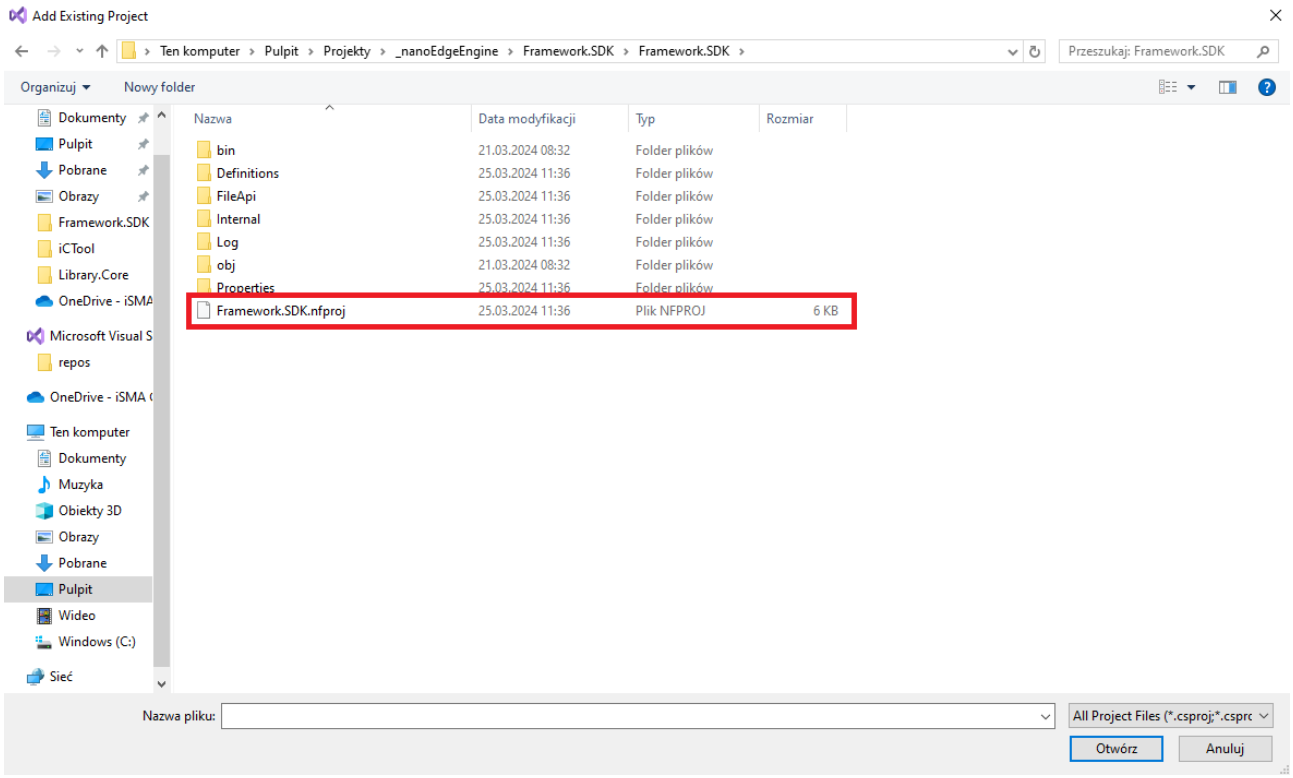


Figure 6. SDK source code location

- Right click on References under the created project and choose Add Reference.

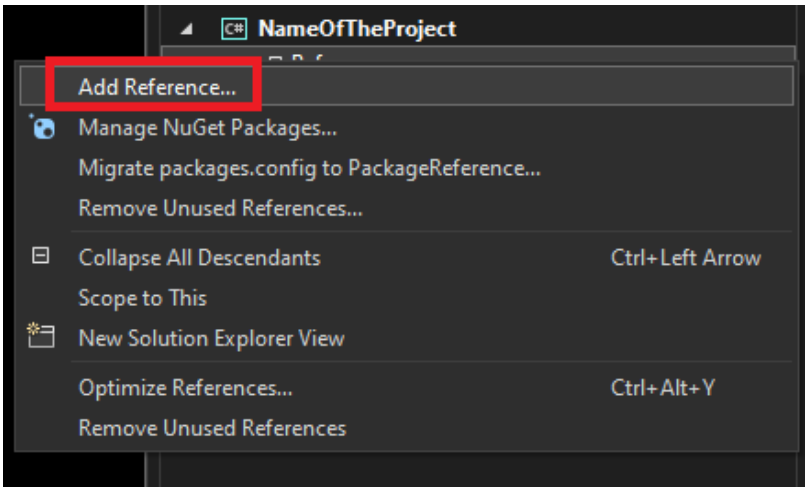


Figure 7. Adding reference option

- In the Reference Manager, choose Projects and select Framework.SDK. Click OK.

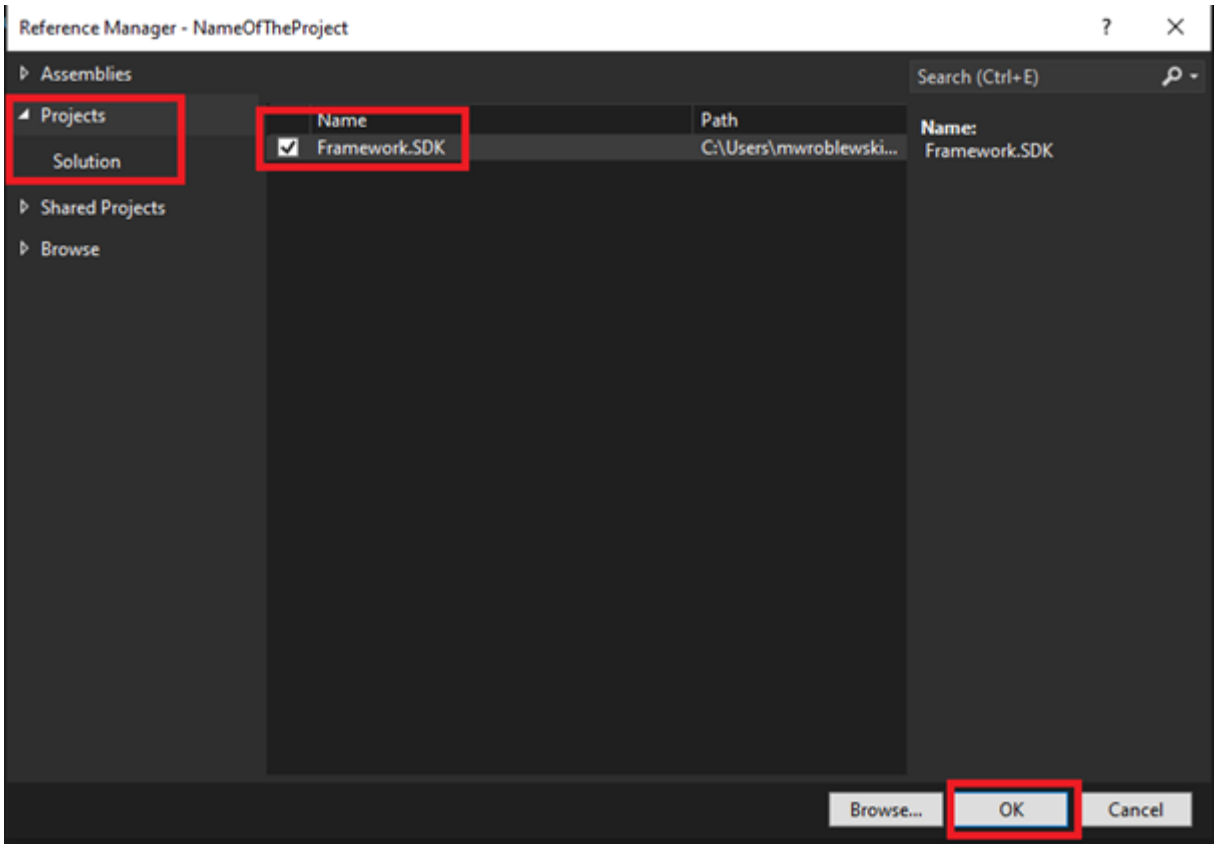


Figure 8. Referencing Framework.SDK

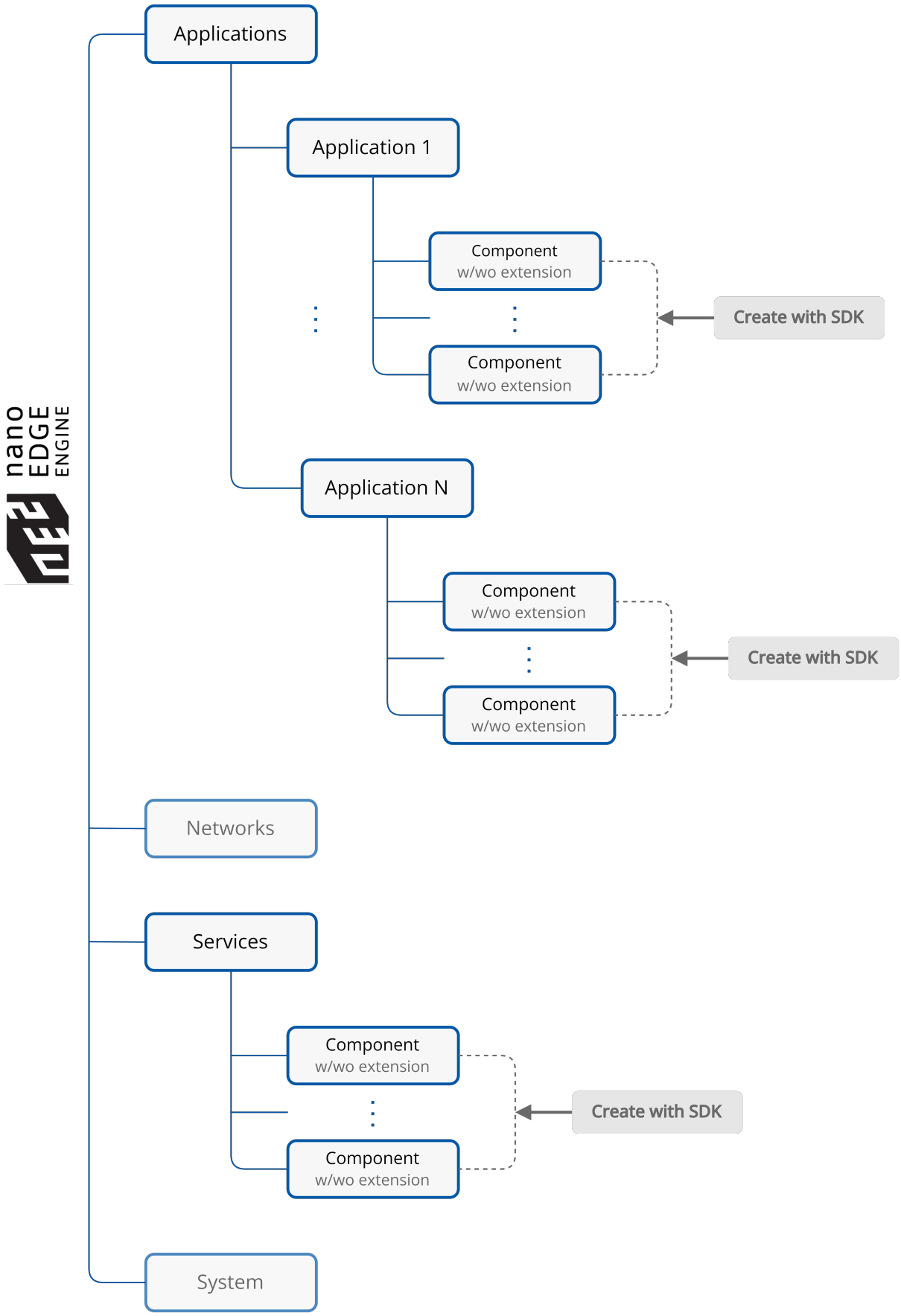
4 Step 3: Programming

The nano EDGE ENGINE SDK is designed to create user-customized libraries with components, which can be implemented directly in the iC Tool's Software Manager and used in applications. The components' functionality may be enhanced by extensions, which can also be created with SDK.

In a nutshell, the nano EDGE ENGINE structure has 3 levels:

- container (Applications, Networks, Services, and System);
    - components (such as Application in the Applications container);
        - components with or without extensions (basic control blocks).

The nano EDGE ENGINE SDK allows the user to create the last level of components, the basic control blocks, which can be located in the Applications or Services containers.


Applications
Application 1
Component
w/wo extension
Component
w/wo extension
Create with SDK
Application N
Component
w/wo extension
Component
w/wo extension
Create with SDK
Networks
Services
Component
w/wo extension
Component
w/wo extension
Create with SDK
System
nano
EDGE
ENGINE

What are the Applications and Services containers?

Applications–a functional center of the device. Here are located Application components, which are designed to build user applications by adding components and linking them into a logical and mathematical sequences. Applications built with the nano EDGE ENGINE are cycle-driven, meaning the included components are executed in repeated periods of time lasting as long as a cycle time is set by the user—thanks to the efficiency of the nano EDGE ENGINE, the cycle may be as short as 100 ms (depending on the complexity of application), allowing effectively real-time calculations. **The nano EDGE ENGINE allows constructing numerous independent, cycle-driven user applications that may work simultaneously**. With the nano EDGE ENGINE the user may freely add as many Application components to the Applications container as needed. The only considerations here are the number of available licensed Data Points, available memory, or cycle time. Data Points are main application components, representing values in applications and allowing to connect services from the Networks container with the application—while connecting components from the Applications and Networks containers, it is recommended to use the Reference linking method. Data Points may be exposed to external networks, which means that external exposition of data from the Applications container or any other space may be carried out only via the Data Point.
Services—a utility center of the device. This container incorporates all services that enhance the basic functionality of the device. As the crude automation data transmission in the **nano EDGE ENGINE** is covered by services in the Networks container, **the Services container provides the value added**: these services cover an exchange of information with systems or devices superior to the building automation level. They may feed the algorithms with data that are normally unavailable in the building automation system, for example, a Weather service, which delivers information (precipitation, humidity, cloudiness, etc.) to the algorithm in the application managing garden watering, or an Alarm service, which sends out alarm messages to external recipients, provided certain limit values are exceeded. These services provide the additional logical layer to algorithms working within applications. They introduce external factors to user applications, enhancing their functionalities outside the core building automation. They may also introduce limit values to evoke certain actions in applications.
Find out more about the nano EDGE ENGINE structure: Basic concept of the nano EDGE ENGINE

4.1 Definitions

4.1.1 Library

The nano EDGE ENGINE library is essentially a folder that contains components. The library created with the nano EDGE ENGINE SDK is user-customized and can be implemented directly into the iC Tool's Software Manager.

Naming recommendations

In theory, libraries created with the nano EDGE ENGINE SDK can be freely named, however, it is strongly recommended to maintain an original naming structure:
Library.Name-1.0.0000.00000.pe
The first part of a naming structure is Library.Name, where "Library" is a constant, and "Name", coming after a dot (.), is a freely definable part. The library name is separated with a hyphen (-) from a version number.
Please note that the **.pe** file extension **is required** for libraries created with the nano EDGE ENGINE SDK.

4.1.2 Component

The nano EDGE ENGINE component is a basic control block, which can be used as part of an application (Applications container) or a service (Services container). Components are linkable.

The component's functionality can be enhanced by extensions.

4.1.3 Extension

The nano EDGE ENGINE component's extension is an enhancement of the component's functionality. By its nature, it is an optional function of a component, which can be switched on and off according to user requirements.

Extensions can be added only to components that have this possibility defined in their code. It is also possible to declare that multiple extensions of the same type can be added to the same component.

4.2 Methods

4.2.1 Create Components

Init()

[protected virtual void Init()]

Init is called, as name suggests, when the component is initialized. It can be used to set up the component before it starts executing.

Execute()

[protected virtual void Execute()]

If a component needs to do something dynamically, like recalculate the output based on inputs or poll data, it can be done using the Execute method.

A components Execute method will be called by its parent service with the service's set frequency. For example, for an Application component with scan time of 200 ms, then every 200 ms the component will go through all of its children in order and call Execute for each of them.

OnRemove()

[protected virtual void OnRemove()]

Called once when the component is removed from device. Can be used for cleanup, unplugging events, etc.

ParentService()

[public ServiceBase ParentService()]

Returns the parent component that the component belongs to.

4.2.2 Create Extensions

Extensions

[public Extension[] Extensions]

List of all extensions attached to component.

AddExtension(Extension newExtension)

[public bool AddExtension(Extension newExtension)]

Adds an extension to component.

RemoveExtension(Extension target)

[public bool RemoveExtension(Extension target)]

Removes the target extension.

RemoveExtension(ushort atIndex)

[public bool RemoveExtension(ushort atIndex)]

Removes the extension at the specified position in the list.

RemoveAllExtensions()

[public bool RemoveAllExtensions()]

Removes all extensions from component.

GetExtension(ushort atIndex)

[public Extension GetExtension(ushort atIndex)]

Returns extension at the specified index.

5 Step 4: Implementation

To implement the user library directly to the Software Manager in the iC Tool, first, compile it in the Visual Studio once all components for the library are ready.

Warning!

Please note that the .pe file extension is required for libraries created with the nano EDGE ENGINE SDK. Libraries compiled in another format will not work properly.

Then, use the Import functionality in the iC Tool (Import>Import Framework Files) and choose the compiled .pe file.

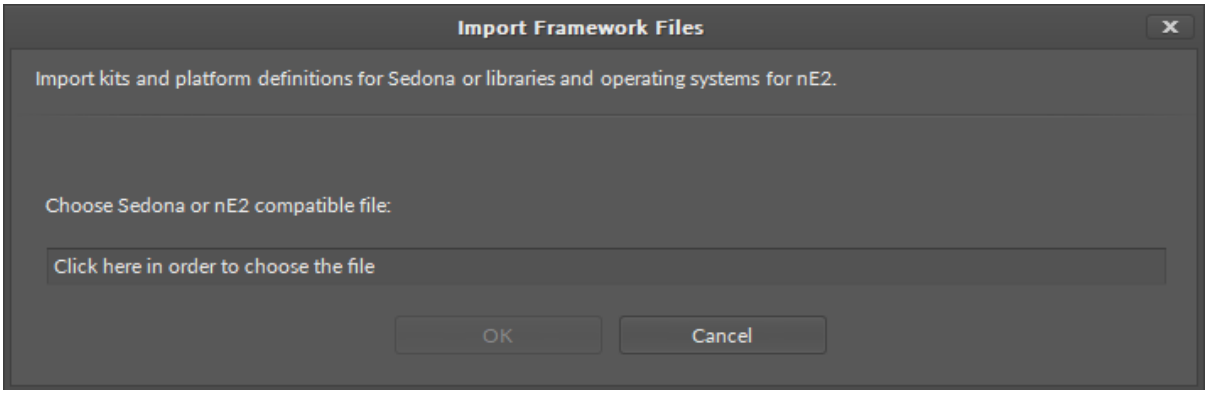


Figure 9. Import Framework Files dialog window in iC Tool

The library will then become visible in the Software Manager ready to be installed on the controller.

To learn how to use the Software Manager, please see the iC Tool user manual.