API documentation

Contents

Introduction

Bentley's ProStructures CONNECT Edition is a very complex and flexible software that allows user to do almost anything they want. However, in certain situations, automating certain tasks or creating a simple extension can drastically increase the user's productivity. Fortunately, this is possible with ProStructures. Any user with basic experience in programming can create their own plugin to suit his or her needs. This document outlines the necessary steps that need to be completed by the user to use the ProStructures API to create his custom plugin. A sample plugin will also be described as an example.

Creating a plugin

Initial setup

Before anything else, the user needs to create a project. In ProStructures, there is an API that serves both ProConcrete and ProSteel and this API is written in both C# and vb.net. The following instructions are for creating a C# plugin using the ProConcrete API and the Visual Studio 2015 IDE.

1) Create a new C# Class Library project. Your project will be built as a DLL.

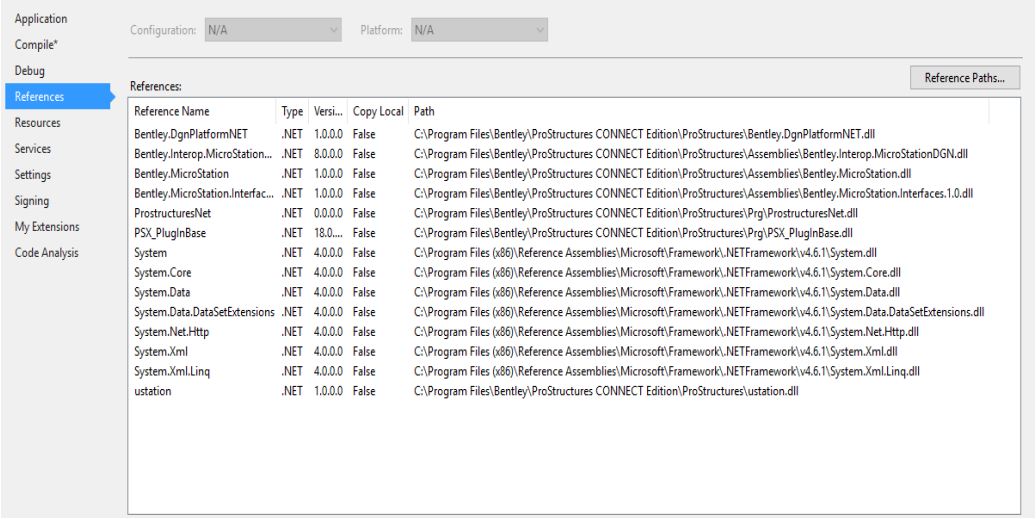
2) Set the properties for your project. It is important that those properties are set correctly for your project to work.

Application tab: Write a name for your assembly. Choose a name that is short and simple if possible as this will make it easier to write the command for the plugin. Once this is done, set the target to .NET framework 4.6.1 to make sure that your project uses the same version as the API. Finally, before going to the next tab, make sure that the application type is set to Class library.

Build tab: Set the target platform to x64 and the build output path to the prg folder of your ProStructures application. By default, the prg folder is located under: C:\Program Files\Bentley\ProStructures CONNECT Edition\ProStructures\prg.

Debug tab: If you want to debug your project, check the "Start external program" radio button and set the path to your ProStructures CONNECT Edition application file. By default, this is located under C:\Program Files\Bentley\ProStructures CONNECT Edition\ProStructures.

References: Add the following references to your project:



Those libraries are needed for your plugin.

After the initial setup

Once the initial setup is completed, you will need to create three new files in your project.

Xml file

First, you will need to create a xml file. This xml file is useful for mapping the command to a function that you will create in your solution. If this is done properly, this function will be called when you type the command in the ProStructures application.

AddInMain.cs

You will also need to create a C# class file that implements the Bentley.MstnPlatformNET. You will provide an id and this id will be used to load your plugin in the application.

.csproj file

You will need to add the following instructions to the .csproj file of your solution to make your xml command file work:

```
<ItemGroup>
  <EmbeddedResource Include="commands.xml">
    <LogicalName>CommandTable.xml</LogicalName>
    <SubType>Designer</SubType>
  </EmbeddedResource>
</ItemGroup>
```

Add those instructions in the <ItemGroup> tag of the file.

Those are all the necessary steps that need to be followed for your plugin to work. You can structure the rest of your solution in any way that suits you.

Note: The steps described above can be confusing. We suggest looking at the sample project provided if some questions arise. This sample is a basic project that can be loaded in the application. You could simply start from this project if you want.

AddInMain.cs

Implements the base AddIn class. This class is identical for every plugin except for the id.

TemplateHandler.cs

This class is responsible for writing to sve, wndcfg and tpl files and reading from them. It contains multiple API function calls related to sve files, wndcfg files and tpl files. The sve and wndcfg files are saved under: C:\ProgramData\Bentley\ProStructures CONNECT Edition\Configuration\ProStructures\Localised\USA_Canada\Temp and the tpl files are saved under: C:\ProgramData\Bentley\ProStructures CONNECT Edition\Configuration\ProStructures\Localised\USA_Canada\Varia

InTransaction.cs

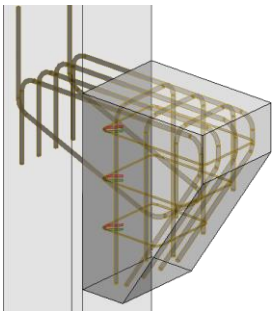
This is a helper class for writing and reading to/from a ProStructures object.

CorbelRebarSet.cs

This class is responsible for building the rebars in the corbels and the support column. The rebars are built according to the parameters in the form and according to the rebar template parameters in the ProStructures application.

Feature Description

In this section, we will cover the corbel distribution plugin's purpose along with its limitations. The procedure that needs to be followed to create a corbel distribution will also be outlined at the end.



Feature summary/GUI

The corbel distribution tool is a very flexible feature that allows users to place multiple corbels on a ProConcrete column or a MicroStation solid. The corbels that are placed are smart solids.

Options Tab

Origin of the distribution: The corbels can be distributed from four different points: from the start of the shape, from the end of the shape, from the middle of the shape and from a user selected point (this can be any point on the shape).

Direction: If the corbels are distributed from the start or from the end, they will always be distributed towards the opposite end of the shape. However, if they are distributed from the middle or from a user selected point, they can be distributed towards the left, the right or in both directions. If the user chooses to distribute from the middle in both directions, the corbels will be equally distributed in both directions. However, if the user chooses to distribute them from a point that is not centered, the corbels will not be equally distributed in both directions. The program will distribute will distribute the most corbels that it can towards the shorter side and the rest of the stiffeners will be distributed in the other direction.

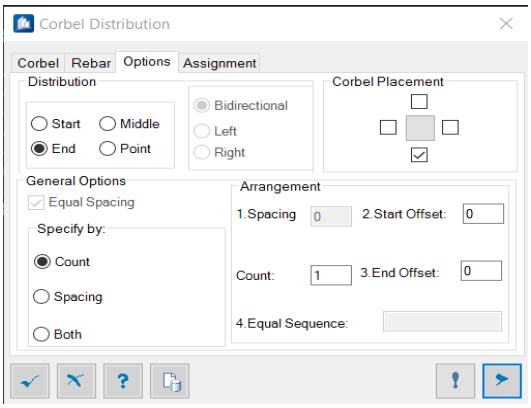
Placement options: The user can specify on which side of the support shape that the corbels will be inserted.

Offsets: The user can specify a start and end offset for his distribution. No corbels will be placed in the areas located between an offset and a shape extremity.

Distribution method: The user can choose to distribute the corbels by specifying a count, a spacing or both. If he chooses the specification by count, the program will calculate the exact spacing that is required between each corbel to fit this exact count and the first and last corbels will be placed on an offset (limits). If the user chooses the specification by spacing, the program will calculate the number of corbels that can fit on the support shape. The first and last corbels will not necessarily be placed on an offset. Finally, if the user chooses the specification by both, he will need to enter a custom sequence or

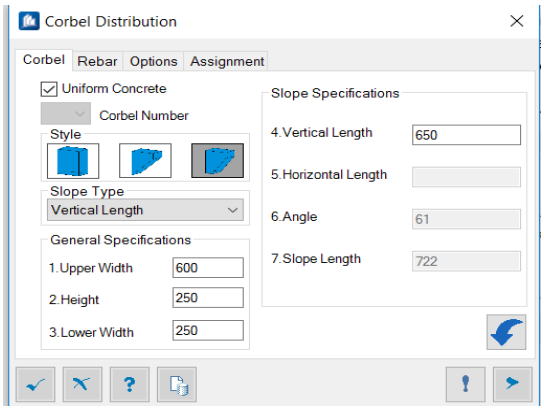
a series of custom sequences depending on whether the equal spacing checkbox is checked. This option allows the user to distribute his corbels in a more flexible way.

Restrictions: There are certain restrictions related to the distribution. First, if the user chooses to distribute corbels by specifying a count and selects a point as the origin of the distribution, he will not be able to distribute the corbels in both directions. Also, if the user chooses to distribute the corbels by both spacing and count and he unchecks the equal spacing checkbox, he will not be able to distribute his corbels in both directions.

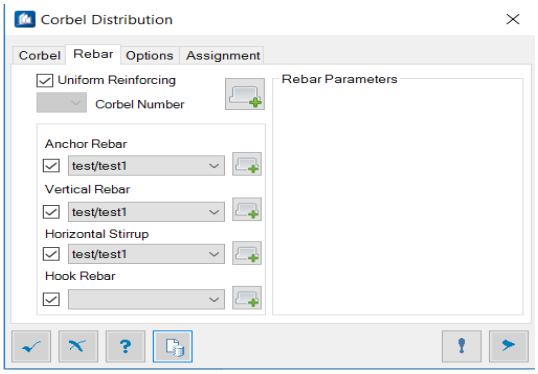


Corbel Tab

In the corbel tab, the user can specify multiple parameters for his corbels such as upper width, height, lower width, and many others. Those parameters alter the appearance of the corbels. These parameters can be unique to every stiffener or they can be identical for all of them. The user can select which corbel to modify by selecting the desired corbel in the upper right combo box and he can check the upper right checkbox to apply the same parameters to each corbel. Finally, there is a reset button located at the bottom right of the tab. When this button is pressed, every parameter will be reset to its default value for the current configuration.

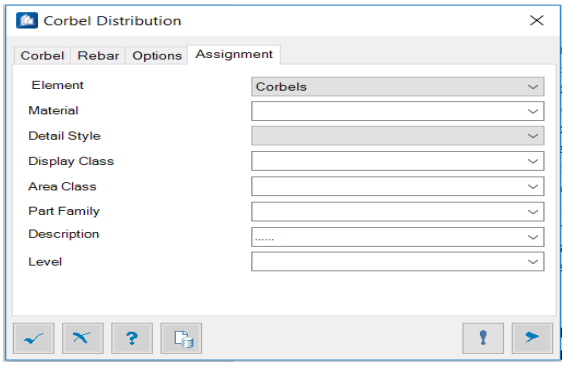


Rebar Tab



In the rebar tab, the user can specify which type of rebar will be added to reinforce his corbels. There are four different types of rebar that can be inserted inside any corbel to reinforce it: anchor rebar, vertical rebar, horizontal stirrup and hook rebar. The user can also specify certain parameters for his rebar. To do so, he will need to click on the buttons with the plus sign located next to the rebar templates combo boxes. When one of these buttons is pressed, controls will appear in the the group box to the right and the user will be able to change the parameters for this specific type of rebar. Finally, like in the corbel tab, the user can choose to specify unique parameters for each stiffener or he can choose to apply the same ones for all of them.

Assignment Tab



In the assignment tab, the user can specify a material for his corbels. He can also assign his corbels to a display class, area class, part family, a level and he can assign a detail style and a description. The user can also assign a level to his rebars and the lines related to those rebars.

Global Buttons

There are multiple buttons located at the bottom of the form. Most of those are buttons that are common to every connection. However, one of the button is responsible for building the corbel distribution with the parameters on the form. The distribution will not appear until this button is pressed.

Usage and workflow

Load the plugin: mdl load PSN_CorbelDistribution

Call the plugin command (once loaded): PSN_CorbelDistribution Create

Call the plugin command without loading it first:

mdl keyin PSN_CorbelDistribution, CORBELDISTRIBUTION PSN_CorbelDistribution create

Creation procedure:

Editing procedure of rebar:

To edit the rebar, double click on any rebar to access the native dialog box.