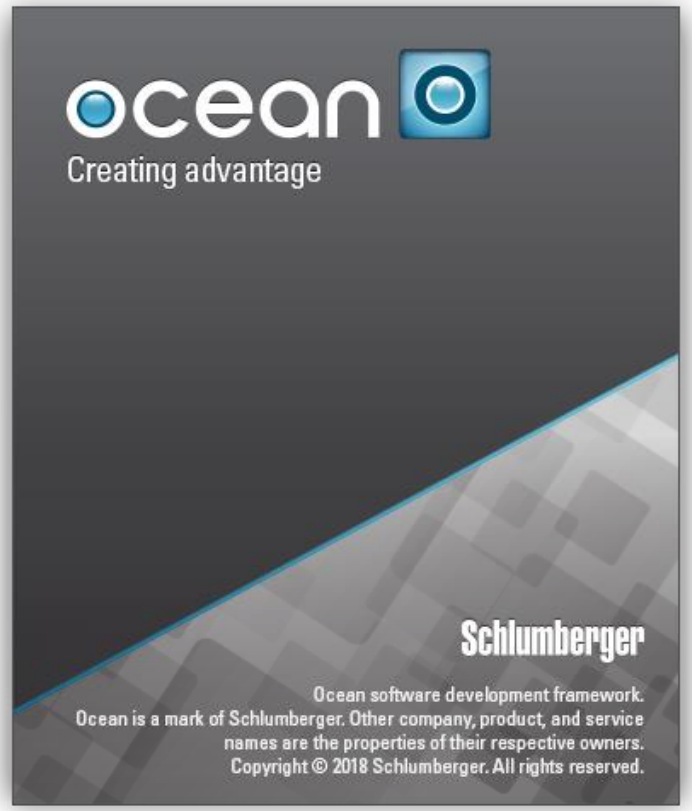
Ocean for Petrel Wizards and Developers Tools in Visual Studio

For Geoscientists and Software Developer



Security Notice

The software described herein is configured to operate with at least the minimum specifications set out by Schlumberger. You are advised that such minimum specifications are merely recommendations and not intended to be limiting to configurations that may be used to operate the software. Similarly, you are advised that the software should be operated in a secure environment whether such software is operated across a network, on a single system and/or on a plurality of systems. It is up to you to configure and maintain your networks and/or system(s) in a secure manner. If you have further questions as to recommendations regarding recommended specifications or security, please feel free to contact your local Schlumberger representative.

TABLE OF CONTENTS

THE OCEAN FOR PETREL WIZARDS

Ocean provides a Wizard to simplify the creation and installation of modules. This chapter provides an overview of the Ocean Module Wizard and its features.

Installing the Ocean for Petrel Wizards

The Ocean for Petrel Wizard is installed as part of the Ocean for Petrel SDK installation process. Installation of the Wizard is an option and requires some disk space. Although it is part of the typical installation, it might not have been part of your installation. If the Wizard has not been installed, you will need to install it before you can go any further.

To install the Wizard:

1. Run your Ocean SDK installation executable again.
2. Select Modify.
3. Add the Wizard using the settings in the associated boxes.

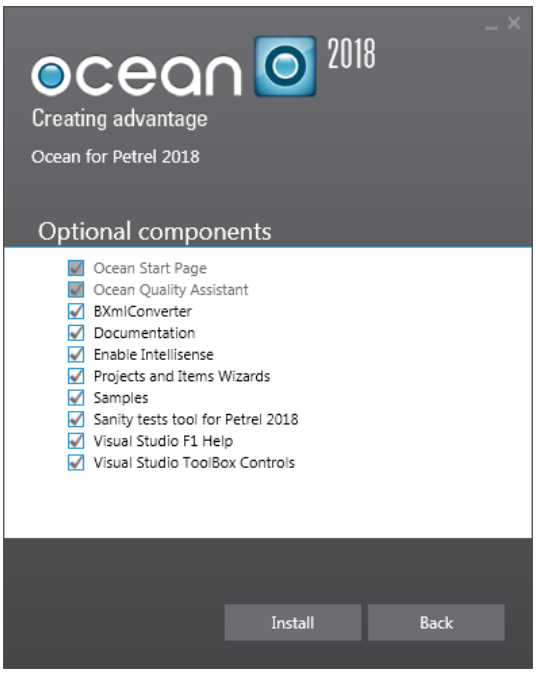


Figure 1: Installing the Ocean Wizard

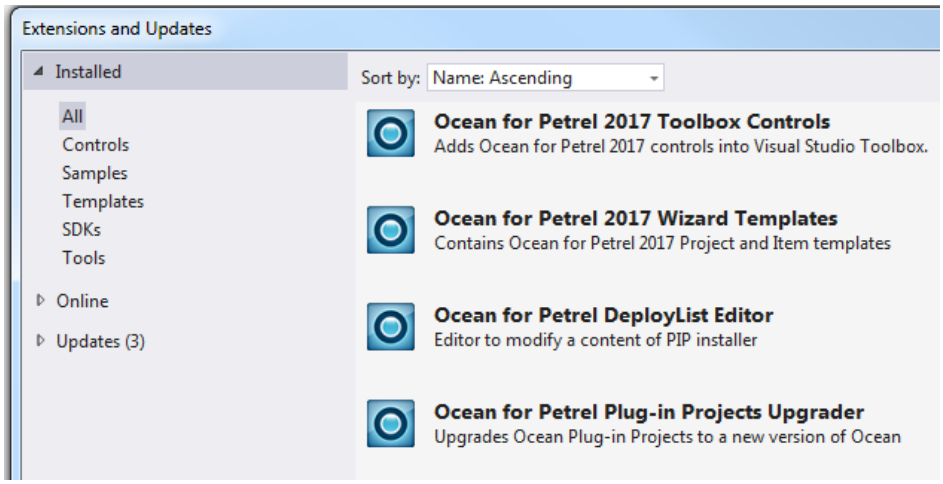


Figure 2: Ocean Wizard developer pack installed

To check whether the Ocean Wizard has been installed, run Visual Studio and select **New Project** from the **File** menu. The project dialog should show **Ocean** as an option in the list of new project types as shown in the following figure.

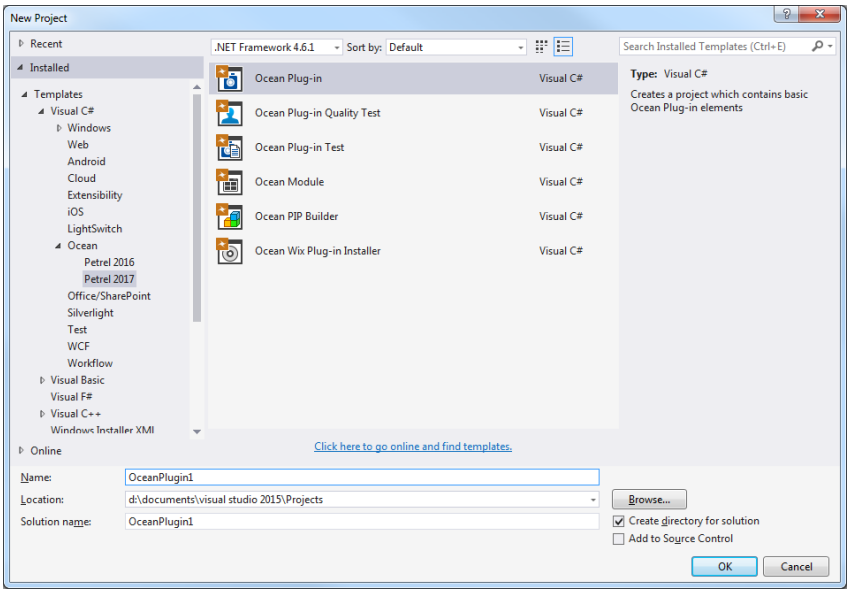


Figure 3: New Project Dialog Showing the Ocean Wizard Options

If these icons are not shown, run the Ocean SDK installer again and select the Wizard installation option.

If you try to add a new project item to an existing project (as shown in Figure 4), then you can see the Ocean project item types (as shown in Figure 5).

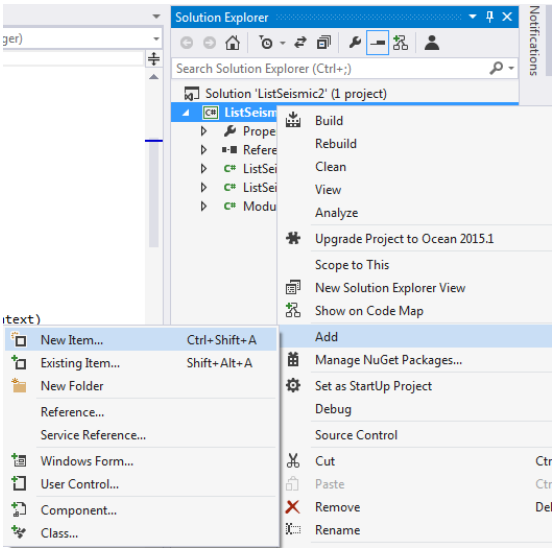


Figure 4: Add / New Item Context Menu on the Solution Explorer

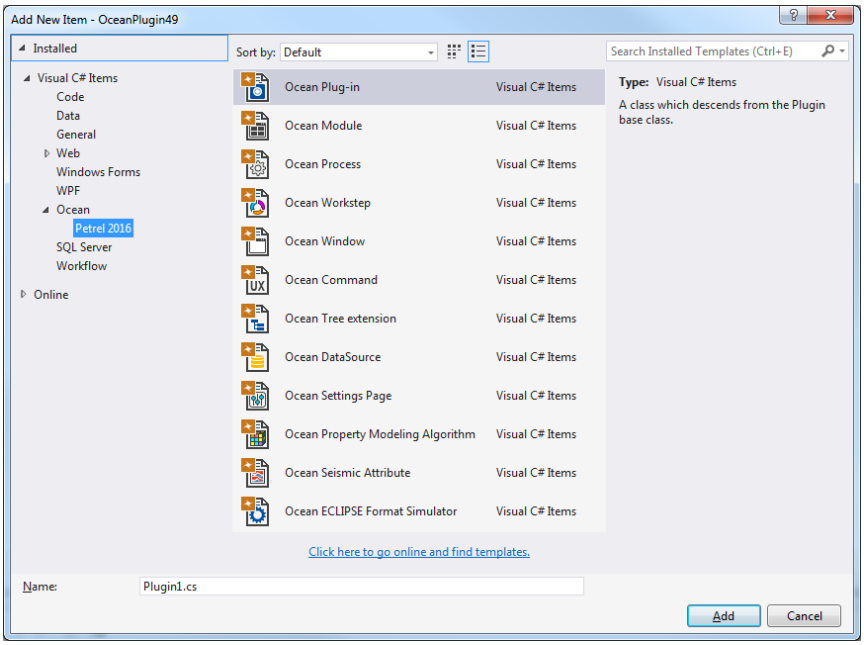


Figure 5: New Project Item Dialog Showing the Ocean Wizard Installed Options

Creating an Ocean for Petrel Plug-in Project

If you want to create a new plug-in project, choose **File** > **New Project** in Visual Studio. The following screen opens.

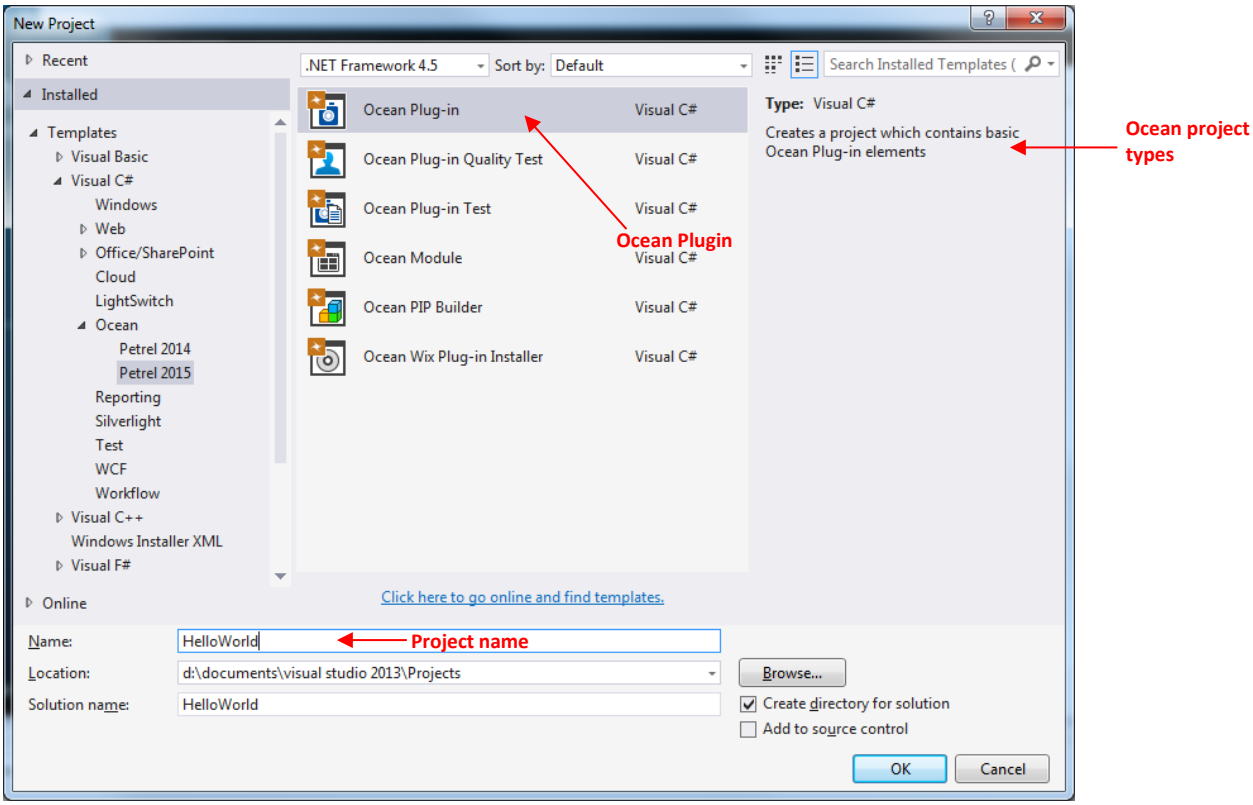


Figure 6: Creation of an Ocean Plug-in Dialog

The following figure shows the Step 1 window of the plug-in project or plug-in file wizard:

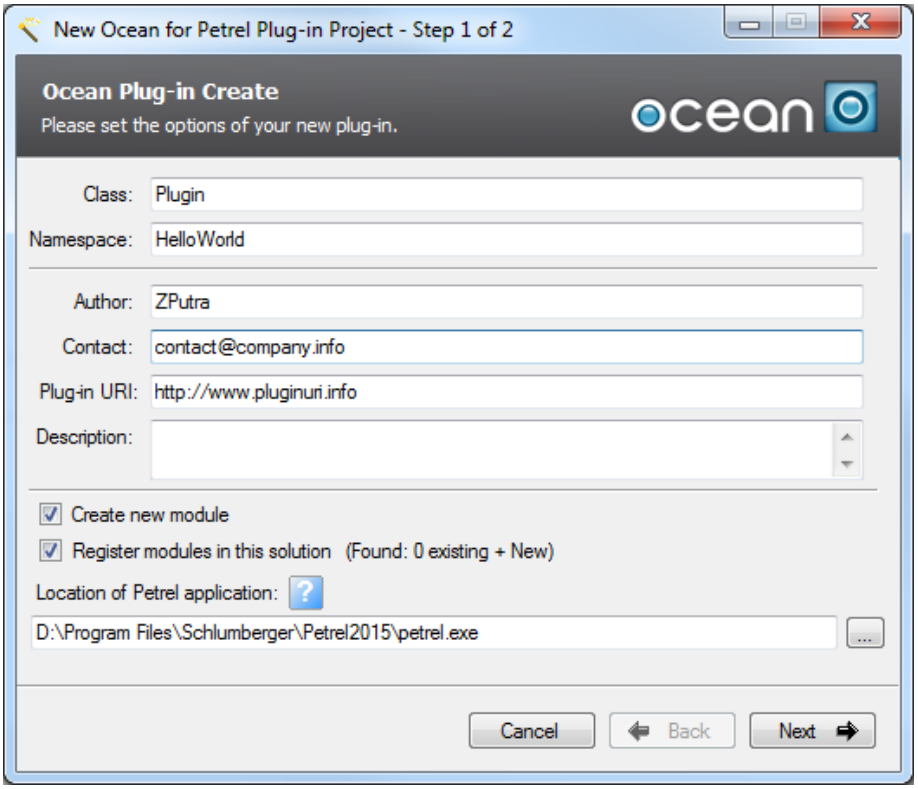


Figure 7: Step 1 of Plug-in Wizard Dialog

In this page, you can specify the plug-in metadata information such as the author, contact information, and a small description. This information is redundant if you use the old IModuleAppearance on your modules, but the information given in the plug-in will be the main source.

If you check the Register existing modules in the solution checkbox, then you can select which modules want to register from the detected module list:

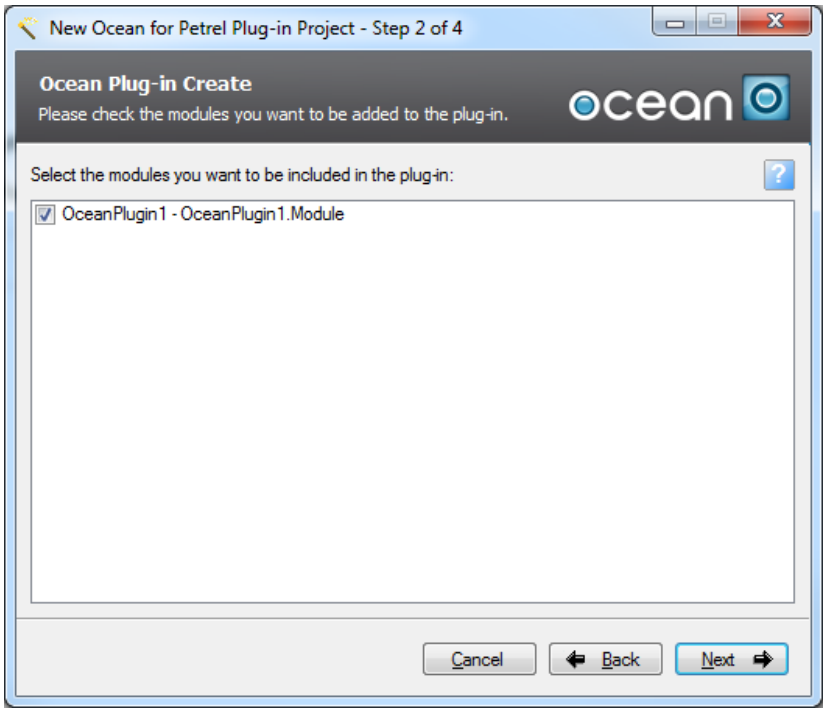


Figure 8: Step 2 of Plug-in Wizard Dialog

In order to use the plug-in, there must be at least one module registered in it.

You also have the option to generate a new module together with the new plug-in. If you choose to generate a new module, you will get the same module-generator wizard pages you are already familiar with.

During the building of your plug-in, as a post build event, the plug-in gets registered in Petrel, so you need to build your plug-in at least once, and after that, the plug-in is usable from Petrel.

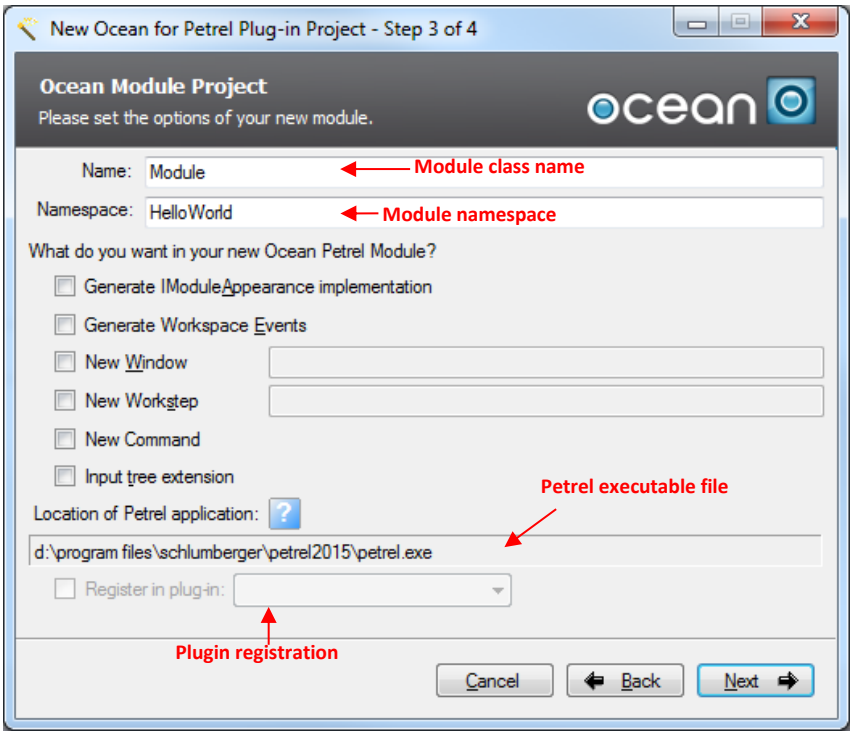


Figure 9: Step 1 of Module Wizard Dialog

To begin creating your new module:

1. Specify the following options:
    - Name: Enter the name of the module class. For this example, change the module name from “OceanModule1” to “Module”.
    - Namespace: Define the namespace for your module project’s class. It is a common practice when defining namespaces to incorporate the company name and product name into the namespace to avoid conflict. In this example, you must change the namespace from “OceanModule1” to “Slb.Ocean.HelloWorldSample”.

- Generate IModuleAppearance implementation: Check this option to generate a class that would implement **IModuleApperance** interface. The Wizard will annotate your Module class with the **ModuleAppearance** attribute. For more information and details of **IModuleAppearance**, refer to the Ocean API documentation or the *Ocean Developers Guide*.
- Generate Workspace Events: Check this option to register for DataManager.Workspace events in the Initialize method of your Module class.
- Location of Petrel Application: Using the browser (…) button, provide the location of petrel.exe file. This will enable the Wizard to set the project configuration to run the Petrel application for instant debugging. Also, the necessary references will be selected relative to this path.

  The following options allow you to add additional ocean items to the project.

- New Window: Check this option to implement and add a custom window to your module project. For more information, refer to Adding a New Window on page 34.
- New Workstep: Check this option to add a new workstep to your module project. For more information, refer to Adding a New Workstep on page 36.
- New Command: Check this option to add a new ribbon to UX mode. For more information, refer to Adding New Command on Page 24. This option replaces New Menu Item option which is still available in Add New Item wizard. For more information, refer to on page 24.

- Input Tree Extension: Check this option to add a new extension to the Petrel tree data items. For more information and details, refer to
- Adding a New Petrel Tree Extension on page 30.

You can also add all of these items to the project later; these checkboxes are only shortcuts. They allow you to create a module with the most usual module elements at once. Their Wizard pages are exactly the same as the standalone Workstep, Menu Item, and Tree Extension Project Item Wizard pages.

2. Click **Next.**

The Step 2 screen of the Wizard dialog opens as shown in the following figure.

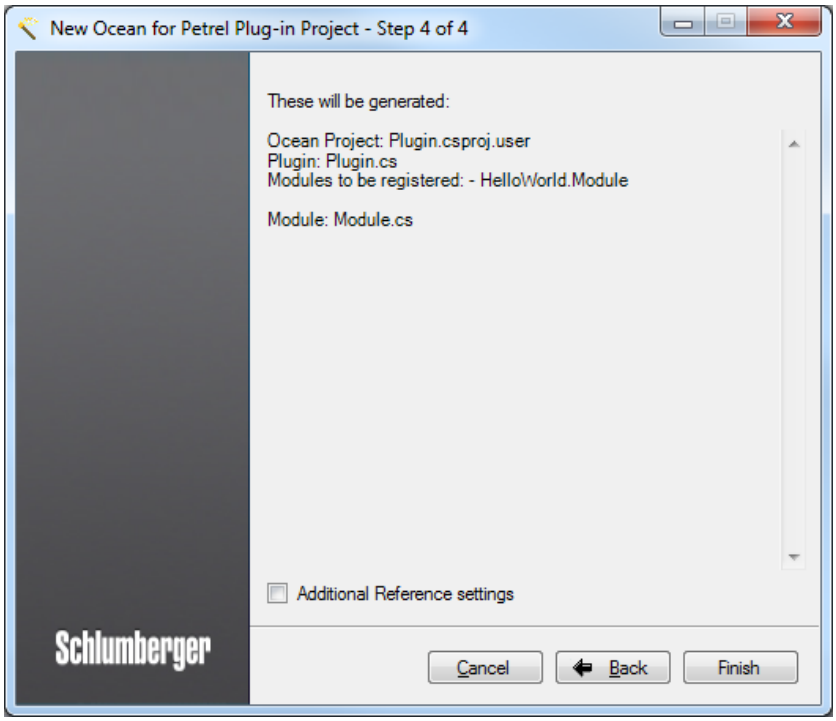


Figure 10: Step 2 of the Wizard Dialog

This step shows a list of the items that will be created. Some of the most important settings are also listed.

3. Click **Finish.**

The Ocean Wizard generates an Ocean Plug-in project with the name provided in the Name textbox (See Figure 6). Ocean Wizard also generates a file with Module class, which implements the **IModule** interface and adds this Ocean module to your new Ocean Plug-in project.

The following figure shows a preview of the Solution Explorer for the newly generated Ocean Plug-in project.

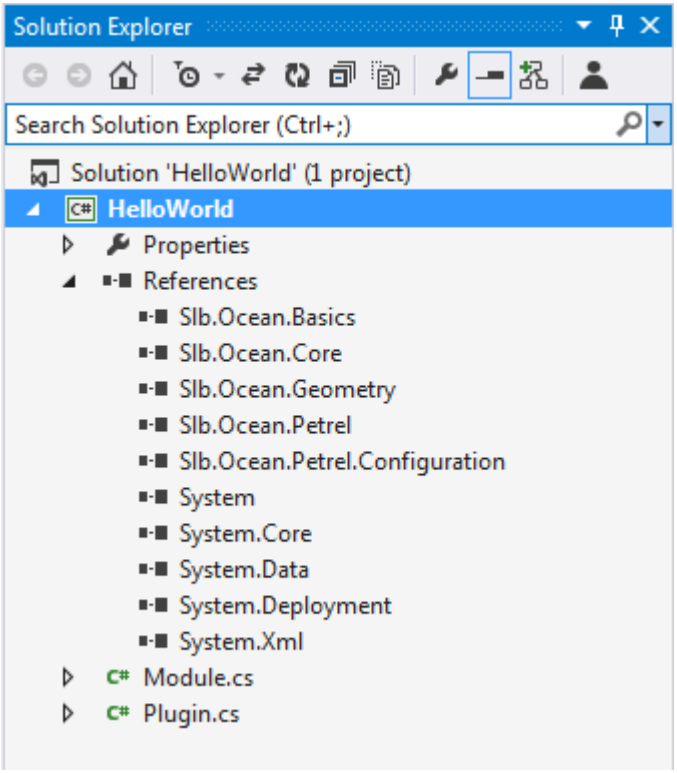


Figure 11: Solution Explorer of the Newly Created Plug-in Project

Adding a New Process

If you chose **Ocean Process** in the **Add / New Item** dialog (Figure 4), then the following screen opens:

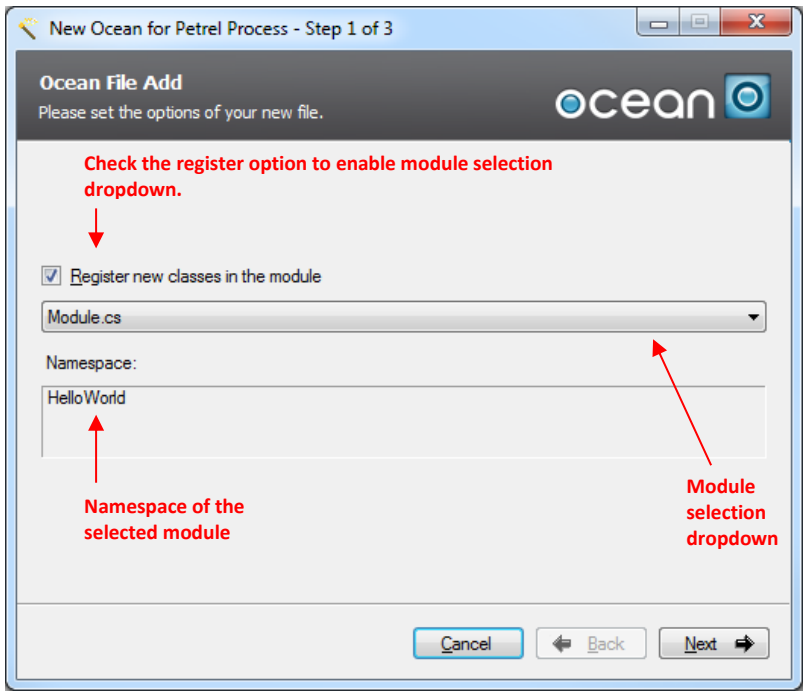


Figure 12: Register in Module Page

1. Specify the appropriate options.

   An Ocean Plug-in assembly may contain more than one module. This page lets you decide where to register your new process or whether to register it at all. The detected modules are listed in the dropdown list.

Important: The same page appears for all Ocean project item types (except the Module itself), and its purpose is the same: to register the generated item in the selected module.

2. Click **Next**.

The following screen opens:

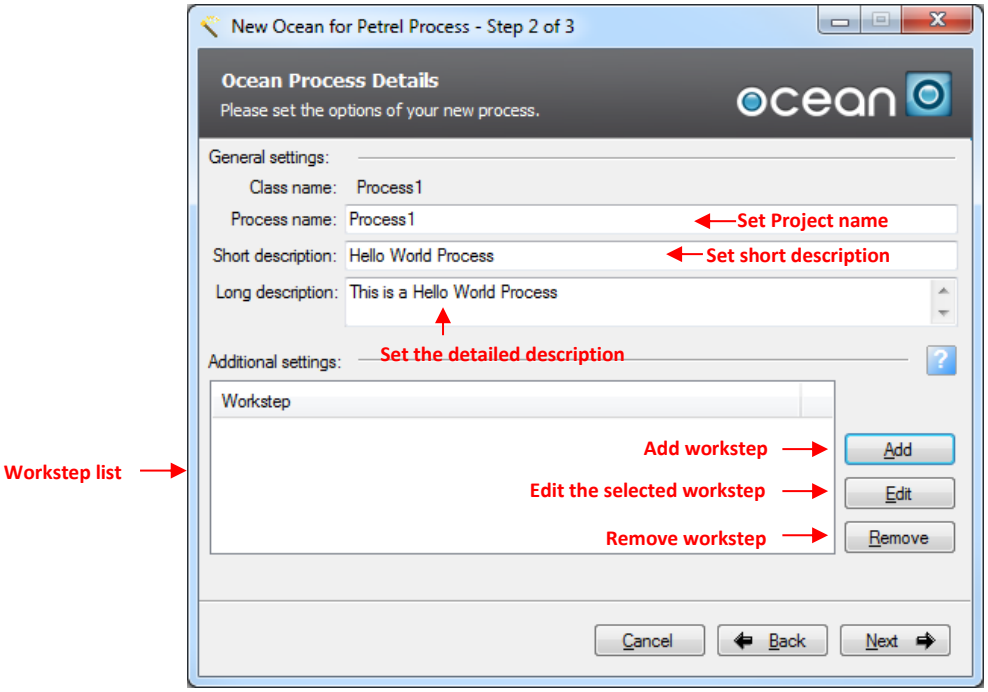


Figure 13: Ocean Process Details

3. Enter the appropriate information in the following fields:
    - Process name: Enter the name of the process class.
    - Short description: Enter a brief explanation of the purpose of the process.

- Long description: Provide a more detailed explanation of what a process does. Your description might include the type of data provided for each of the arguments.

4. Click **Add** to add a workstep. It will open the Workstep Generator dialog as shown in the following figure:

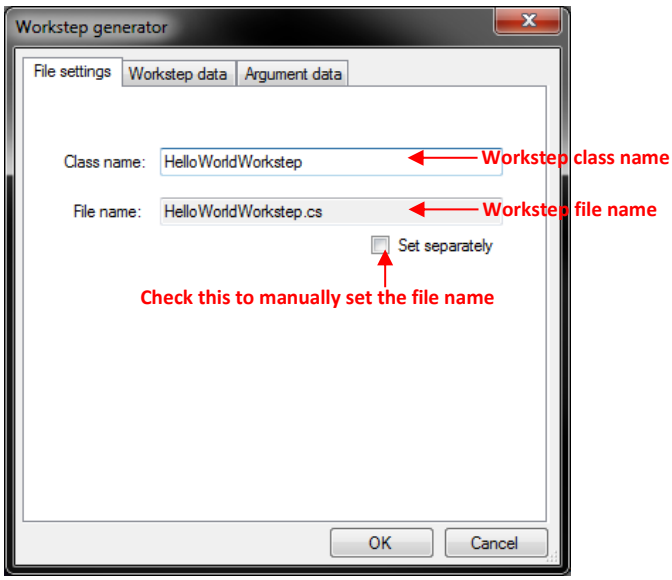


Figure 14: Ocean Workstep Generator

The Workstep Generator dialog contains the following three tabs:

- **File settings**: The worksteps are created into different files, and each workstep has its own source file. On this tab, you can set the file name and class name of the workstep. For detailed instructions, see Adding a New Workstep on page 36.

- **Workstep data**: This tab is similar to the main page of the process. You can set the name, description, and other settings. This tab is exactly the same as on the second page of the Ocean Workstep project item wizard; see Adding a New Workstep on page 36. for details.
- **Argument data**: This tab is used for setting the arguments of the workstep. See Adding a New Workstep on page 36, because it is exactly the same as the third page of the Ocean Workstep project item wizard.

5. After you set the workstep, click **OK** to return to Ocean Process Details.

   In this example, the newly created “HelloWorldWorkstep” is added to the workstep list as shown in the following figure.

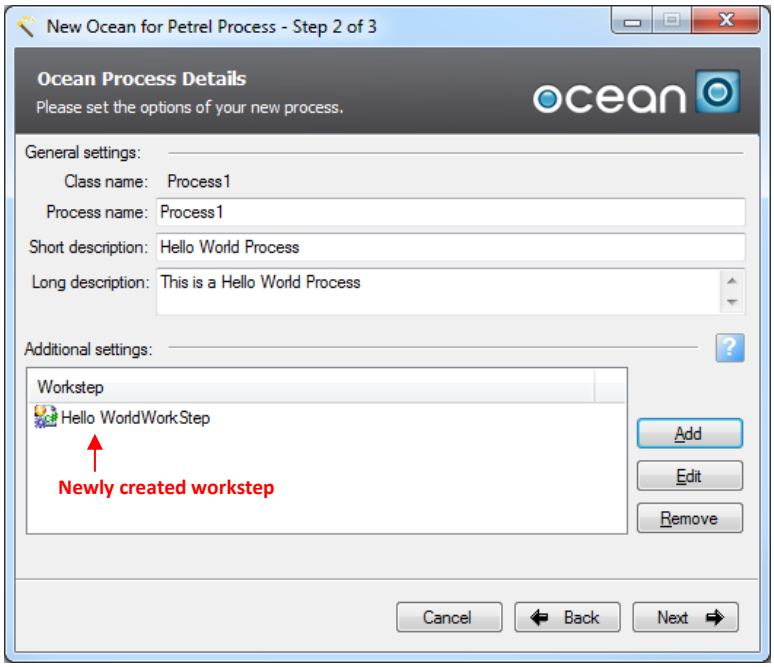


Figure 15: Creation of a New Workstep

6. Click **Next**.

The Wizard generates a file with the Process class and a file with the Workstep class and adds them to the process diagram and workflow editor respectively. The Wizard also adds the following code in the Integrate method of your Ocean module class:

```
public void Integrate()
{
    // Registrations:

    // TODO:  Add Module.Integrate implementation

    // Register HelloWorldProcess
    HelloWorldProcess helloworldprocessInstance = new
HelloWorldProcess();
    PetrelSystem.ProcessDiagram.Add(helloworldprocessInstance,
"Plug-ins");

    // Register HelloWorldWorkstep
    HelloWorldWorkstep helloworldworkstepInstance = new
HelloWorldWorkstep();

PetrelSystem.WorkflowEditor.Add(helloworldworkstepInstance);
    PetrelSystem.ProcessDiagram.Add(new
Slb.Ocean.Petrel.Workflow.WorkstepProcessWrapper(helloworldwor
kstepInstance), "Plug-ins");
}
```

The following figure shows a preview of the process as well as the workstep files in the Ocean Plug-in project.

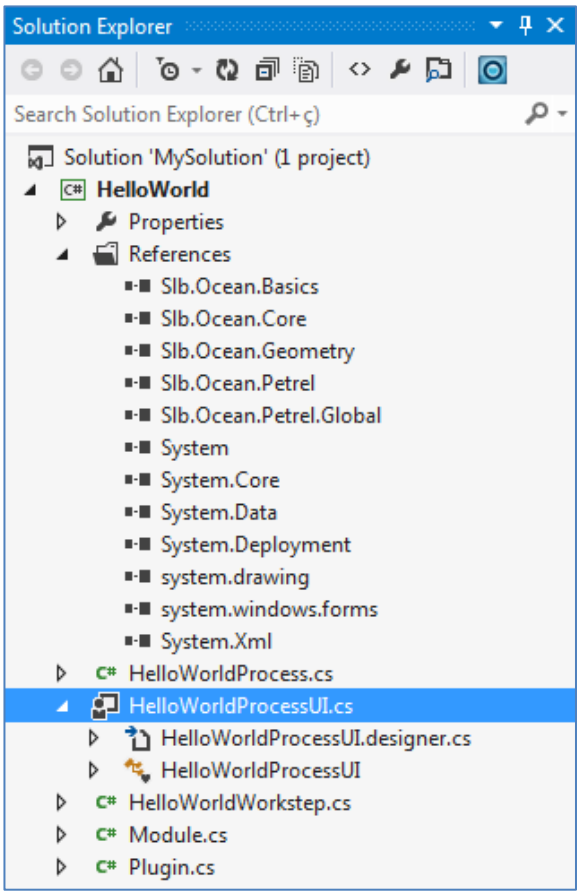


Figure 16: Preview of Solution Explorer with Process and Workstep Files

Adding New Command

To add your own button to Petrel ribbon UI, chose **Ocean Command** from the **Add/New Item** option from the Visual Studio context menu, or if you checked the option to add a new command in Step 1 of the Wizard dialog, then you will see the following wizard screen:

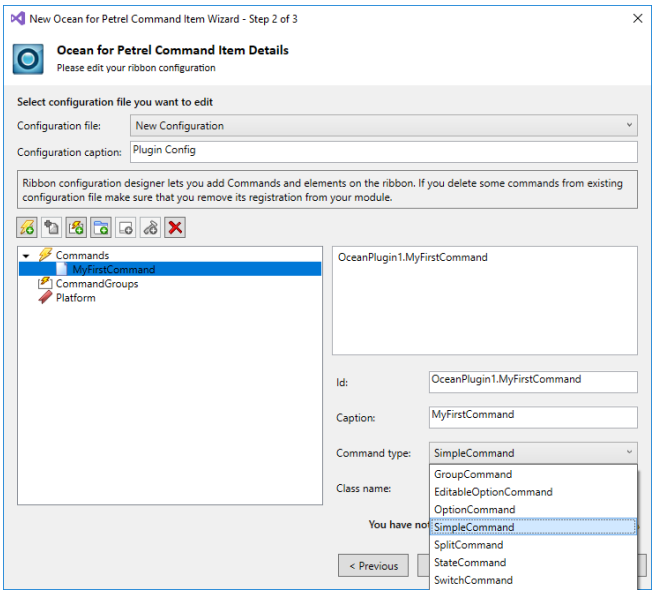


Figure 17: Preview of Solution Explorer with New Command resources

From this screen, you can add new command, create new ribbon or ribbon tab, or extend existing Petrel command group with commands. You must always select the parent item to which the new item will be added.

The types of the menu items that can be added include the following, this wizard will create basic container and register these commands to make them visible in the ribbon. For detail guide and sample on how to implement your logic for these command refer to Ocean for Petrel CHM under Slb.Ocean.Petrel.Commands namespace:

SimpleCommand: Simple command to perform a specific action. When added to Petrel UI it will be rendered as a basic button.

StateCommand: Command that indicates one of two states (e.g. "on" or "off"). When added to the Petrel UI, this will be represented by a button with an on/off state.

TextCommand: Command that defines an editable text option. When added to the Petrel UI, this will be represented by a Textbox.

OptionCommand: Command that defines several selectable options. When added to the Petrel UI, this will be represented by a drop down list.

EditableOptionCommand: Command that defines several selectable options with the possibility to type other texts as well. When added to the Petrel UI, this will be represented by a Dropdown list with an editable text area.

GroupCommand: Button with the appearance of the GroupCommand. When clicked popup menu list of sub-commands is displayed.

SwitchCommand: Button with the appearance of the GroupCommand. When clicked popup menu list of sub-commands is displayed.

SplitCommand: Split button. The primary part displays and executes the selected sub-command while the secondary part display a popup menu list of sub-commands.

To create a new command, select Commands parent item in the tree and click “Add New Command”. Command id, caption and type can be change as shown in below:

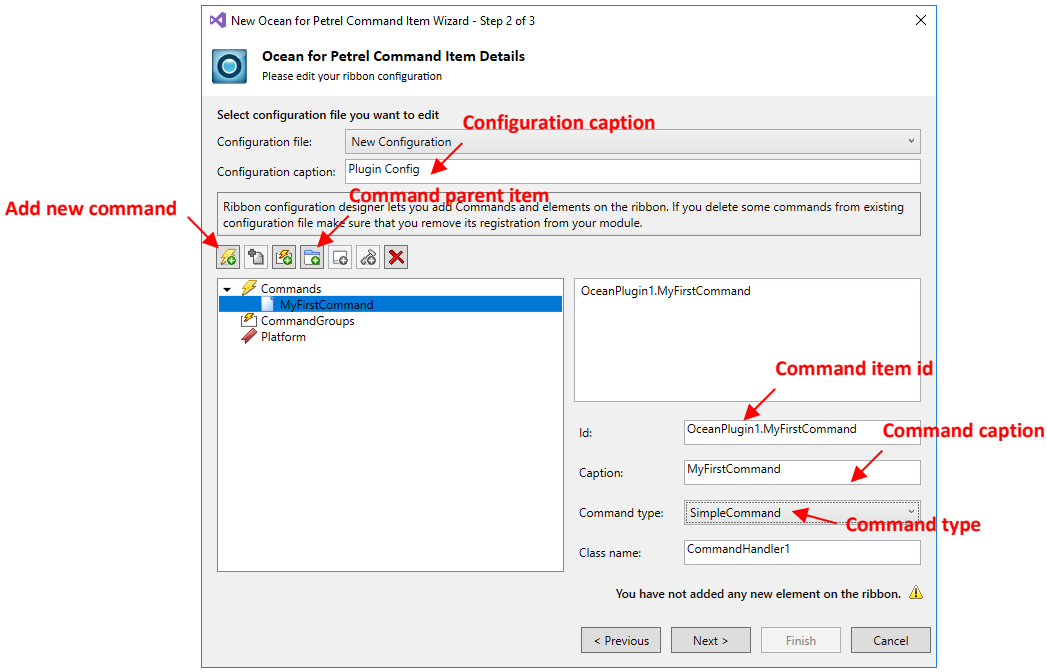


Figure 18: Wizard to add command

To include the new command you created above, select ribbon parent item and click “Add new ribbon to tab”. Then click “Add new ribbon tabgroup”. Finally to add your command to tabgroup, click “Add command to tabgroup” and your ribbon tree will apear as below:

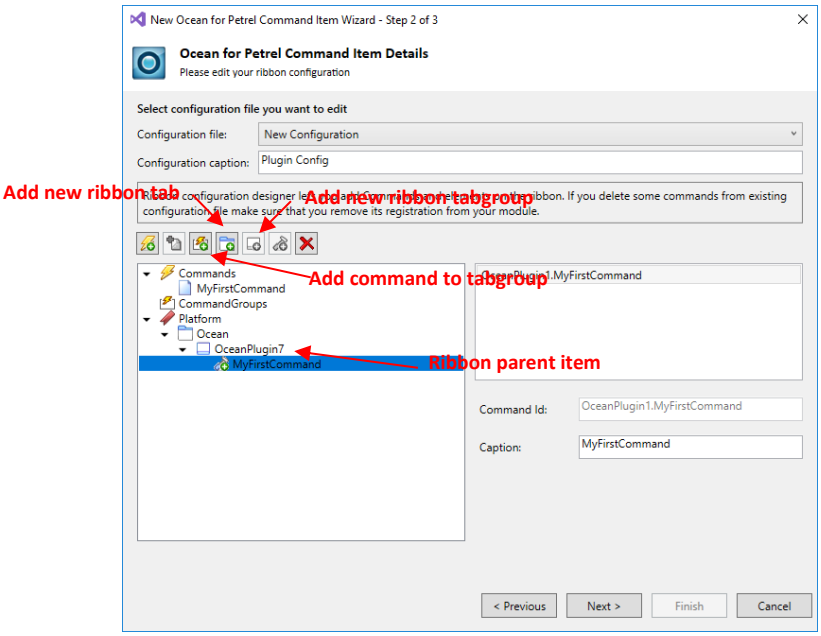


Figure 19: Wizard to add command to ribbon

Once plugin is installed, developer can also manage more advance command appearance properties and ribbon settings using configuration designer in Petrel. Launch configuration designer use CTRL + F2. Configuration designer is available with Ocean Marketing License. Refer to Quick Start for Ocean UX white paper in Ocean for Petrel CHM for detail on how to use configuration designer.

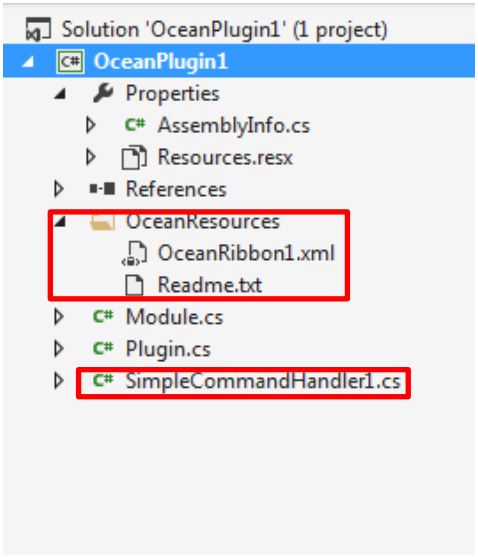


Figure 20: Preview of Solution Explorer with New Command resources

OceanResources folder which will contain OceanRibbon.xml and Readme.txt . OceanRibbon.xml is also registered as resources of the plugin-in project (Figure 21)

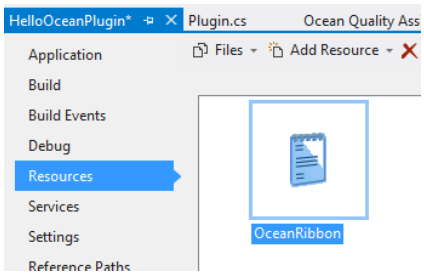


Figure 21: OceanRibbon.xml registed as plug-in project resources

- <CommandName>handler.cs implements SimpleCommandHandler. The default implementation will output to Petrel logger window when command button is clicked.

```
class MyFirstCommandHandler : SimpleCommandHandler
{
    public static string ID = "OceanPlugin7.MyFirstCommand";

    #region SimpleCommandHandler Members

    public override bool CanExecute(Slb.Ocean.Petrel.Contexts.Context context)
    {
        return true;
    }

    public override void Execute(Slb.Ocean.Petrel.Contexts.Context context)
    {
        //TODO: Add command execution logic here
        PetrelLogger.InfoOutputWindow(string.Format("{0} clicked", @"MyFirstCommand" ));
    }

    #endregion
}
```

Figure 22: Command Handler implementation.

Output in Petrel will be ribbon labelled Ocean, ribbon tab and a command. Note: In this example the command created is a simple command

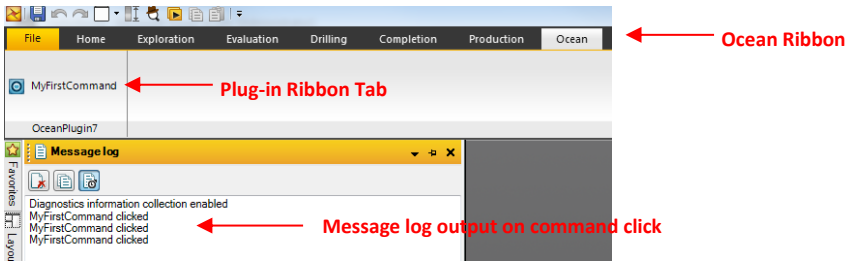


Figure 23: OceanRibbon.xml registered as plug-in project resources

Adding a New Petrel Tree Extension

If you chose **Ocean Tree extension** from the **Add/New Item** option from the Visual Studio context menu (Figure 4), or if you checked the **Input tree extension** option in Step 1 of the Wizard dialog (Figure 21), then you will see the following Wizard screen:

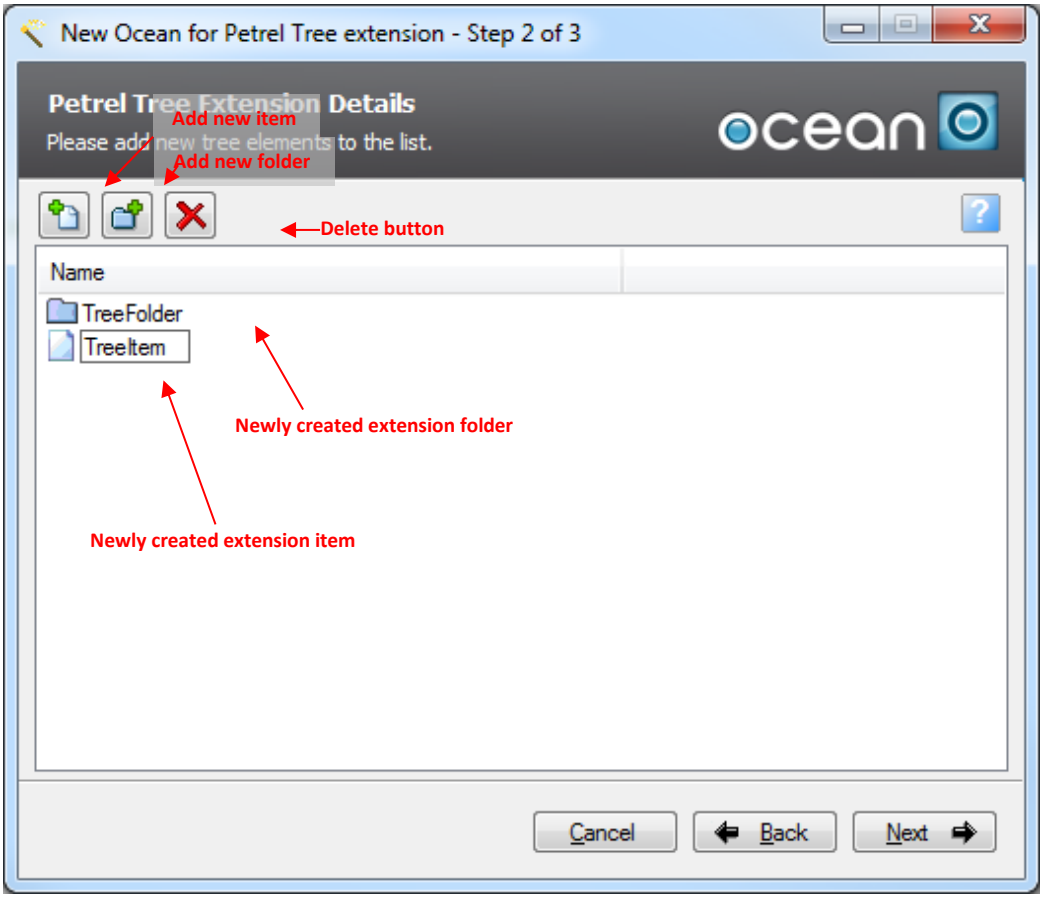


Figure 24: Ocean Tree Extension Details

1. Click the Add Extension button or Add Extension Folder button.
2. Enter the appropriate information.

   For every tree extension, the Wizard generates a file with the tree item class and a file with tree item factory class, files with name and info classes. In this example, the Wizard will generate the following files:

   - HelloWorldTreeitem.cs
   - HelloWorldTreeitemFactory.cs
   - HelloWorldTreeitemImageInfo.cs
   - HelloWorldTreeitemNameInfo.cs

   For every tree extension folder, the Wizard generates a file with tree folder class, a file with factory class, and two files with image and name info classes. In this example, the Wizard will generate the following files:

   - HelloWorldTreeFolder.cs
   - HelloWorldTreeFolderFactory.cs
   - HelloWorldTreeFolderImageInfo.cs
   - HelloWorldTreeFolderNameInfo.cs

3. Click **Next**.

The following figure shows a preview of files created by the Wizard in Solution Explorer.

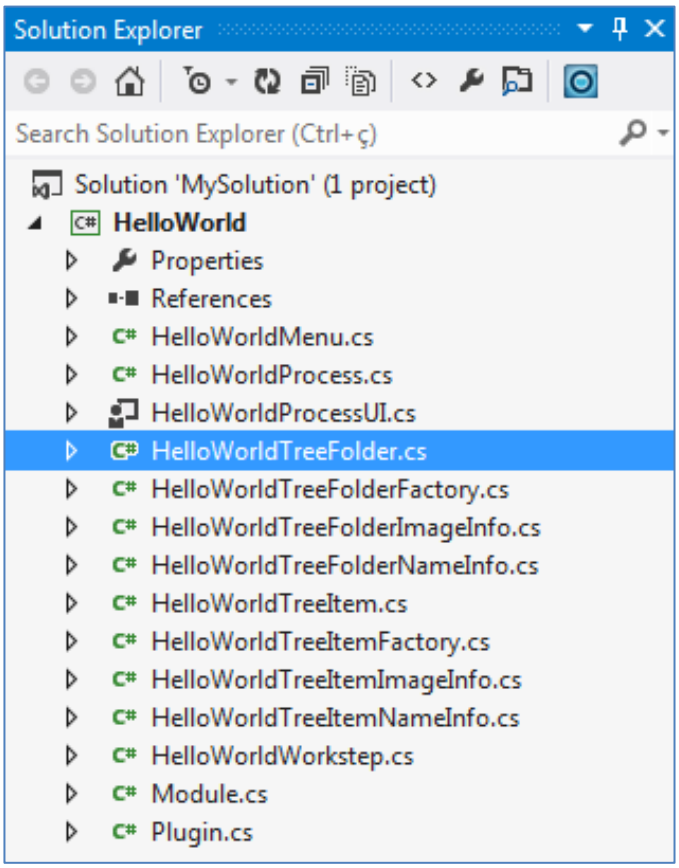


Figure 25: Tree Extension Files in Solution Explorer

In the module file, the **Integrate** method will be extended with the following lines:

```
public void Integrate()
{
    // Registrations:
```

```
    // TODO:  Add Module.Integrate implementation

CoreSystem.Services.AddService(typeof(HelloWorldTreefolder),
typeof(Slb.Ocean.Petrel.UI.INameInfoFactory),
HelloWorldTreefolderFactory.Instance);

CoreSystem.Services.AddService(typeof(HelloWorldTreefolder),
typeof(Slb.Ocean.Petrel.UI.IImageInfoFactory),
HelloWorldTreefolderFactory.Instance);
    CoreSystem.Services.AddService(typeof(HelloWorldTreeitem),
typeof(Slb.Ocean.Petrel.UI.INameInfoFactory),
HelloWorldTreeitemFactory.Instance);
    CoreSystem.Services.AddService(typeof(HelloWorldTreeitem),
typeof(Slb.Ocean.Petrel.UI.IImageInfoFactory),
HelloWorldTreeitemFactory.Instance);
}
```

Adding a New Window

If you chose **Ocean Window** from the **Add/New Item** option from the Visual Studio context menu (Figure 4), or if you checked the option to add a window in Step 1 the Wizard dialog (Figure 21), then the Wizard generates a file with Window class. In addition to the class file, Ocean Wizard also generates a default UserControl with the same name as the Window field (Figure 21) and appended by 'UI' as the class name.

In this example, you entered "HelloWorldWindow" in the "New Window" field (Figure 21), and the wizard generated the following files:

- HelloWorldWindow.cs: contains the Window class which implements IImageInfoSource, INameInfoSource interfaces and inherits from ToggleWindow base class.
- HelloWorldWindowImageInfo.cs: contains ImageInfo class.
- HelloWorldWindowNameInfo.cs: contains NameInfo class.
- HelloWorldWindowUI.cs: contains UserControl class.

The following figure shows a preview of files created by the Wizard in Solution Explorer.

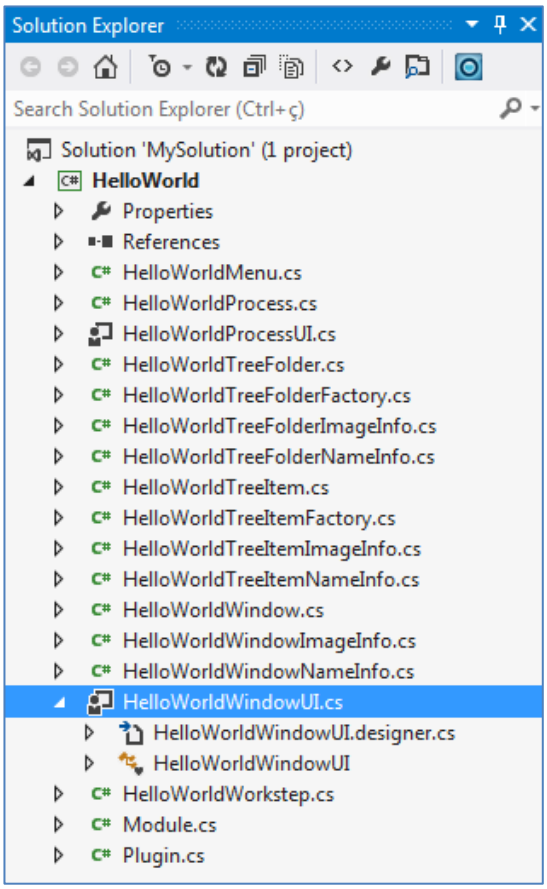


Figure 26: Preview of Solution Explorer Containing Window Class Files

Adding a New Workstep

If you chose **Ocean Workstep** from the **Add/New Item** option from the Visual Studio context menu (Figure 4), or if you checked the option to add a window in Step 1 of the Wizard dialog (Figure 21), then you will see the following Wizard page.

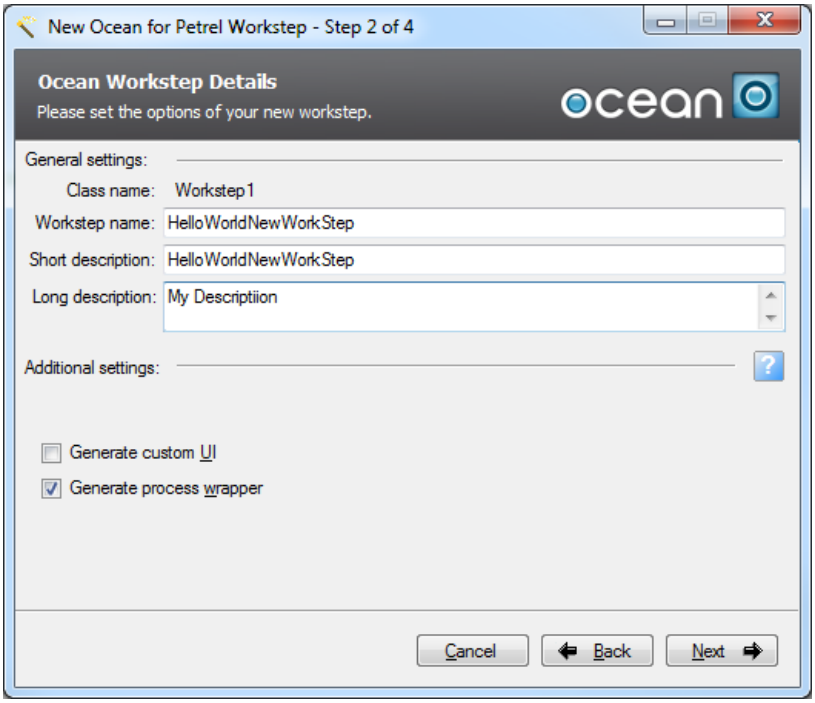


Figure 27: Second Page of the Ocean Workstep Wizard

This page is similar to the general settings page of the Ocean process (see Page 18).

1. Enter the appropriate information.
   The name will be displayed in the process tree of Petrel, and the descriptions will be displayed on the default process UI.

2. Click Next.
3. On the next page (shown below), set the arguments of the workstep. These arguments are input and output variables that will be used by the workstep during its work.

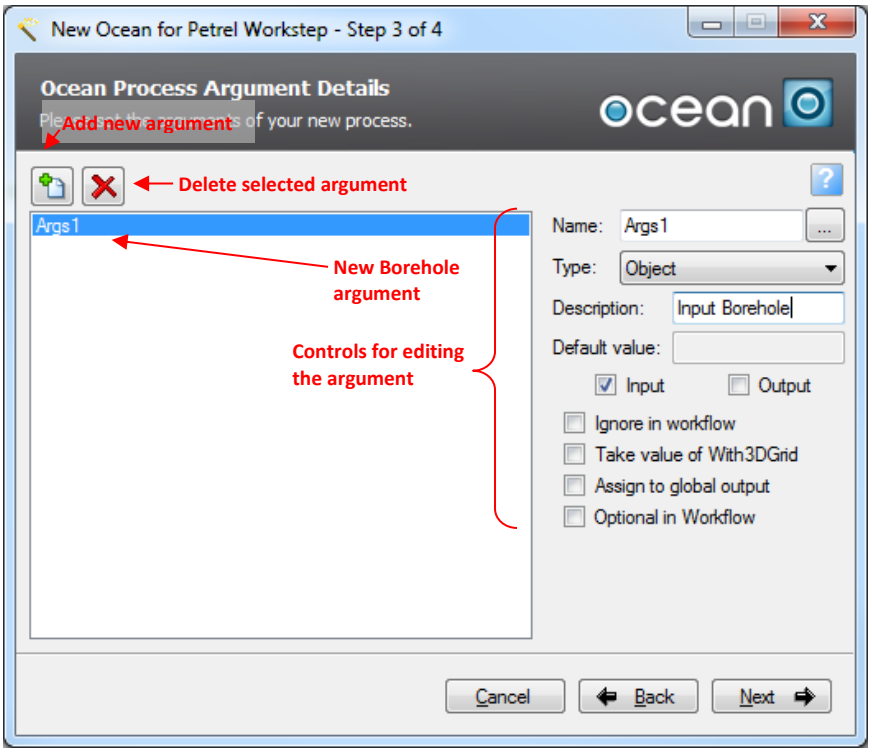


Figure 28: Workstep Argument Setting Page of the Wizard

The Wizard adds the following code in the **Integrate** method of the selected module class:

```
public void Integrate()
{
    // Registrations:

    // TODO:  Add Module.Integrate implementation

    // Register HelloWorldWorkstep
    HelloWorldWorkstep helloworldworkstepInstance = new
HelloWorldWorkstep();

PetrelSystem.WorkflowEditor.Add(helloworldworkstepInstance);
    PetrelSystem.ProcessDiagram.Add(new
Slb.Ocean.Petrel.Workflow.WorkstepProcessWrapper(helloworldwor
kstepInstance), "Plug-ins");
}
```

Adding an Ocean Settings Page

You can use the Wizard to generate custom Settings Pages by choosing **Ocean Settings Page** on the **Add New Item** dialog of Visual Studio (Figure 5). After completing the Wizard steps, you will see the following files:

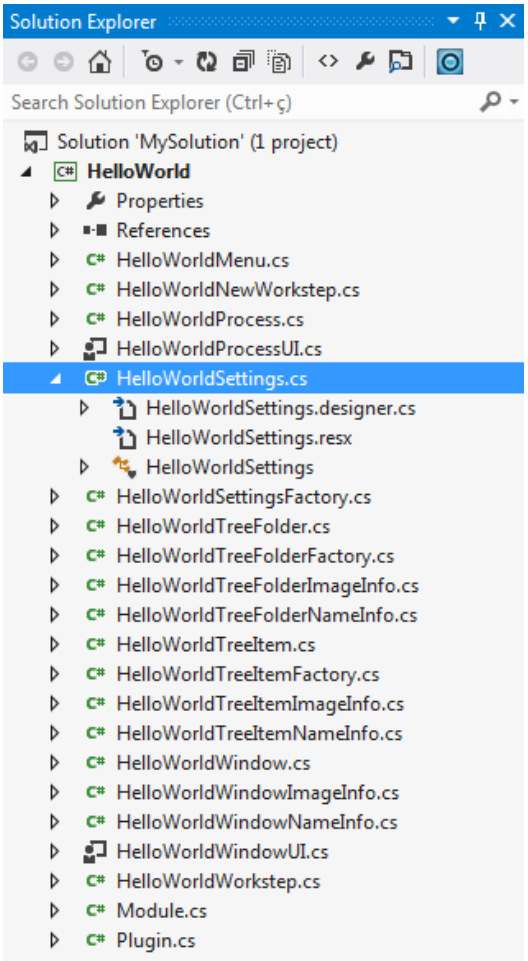


Figure 29: Ocean Settings Page in the Solution Explorer

The Wizard generates a **UserControl**, which will be the SettingPage class, and a SettingPageFactory class.

In this example, you provided the default “DialogPage1” as the name in the Visual Studio Item dialog, and the Wizard generated the following files:

- HelloWorldDialogPage.cs
- HelloWorldDialogPageFactory.cs

If you checked the **Register new classes in module** option in the Visual Studio Item dialog (Figure 12), then the Wizard adds the following code in the **IntegratePresentation** method of the selected module class.

```
public void IntegratePresentation()
{
    // Registrations:

    // TODO:  Add Module.IntegratePresentation implementation
    PetrelSystem.DialogBuilder.AddFactory(new
HelloWorldDialogPageFactory());
}
```

Adding Ocean Data Sources

The Ocean Wizard allows you to add custom data sources or the recommended structured archived data source only to existing Ocean module projects. (See Figure 4 and Figure 5.). Refer to the *Ocean Developers Guide* and *CHM* documentation on *Managed Persistence.*

The first option to add framework to handle structured archive data source with option to create custom domain object.

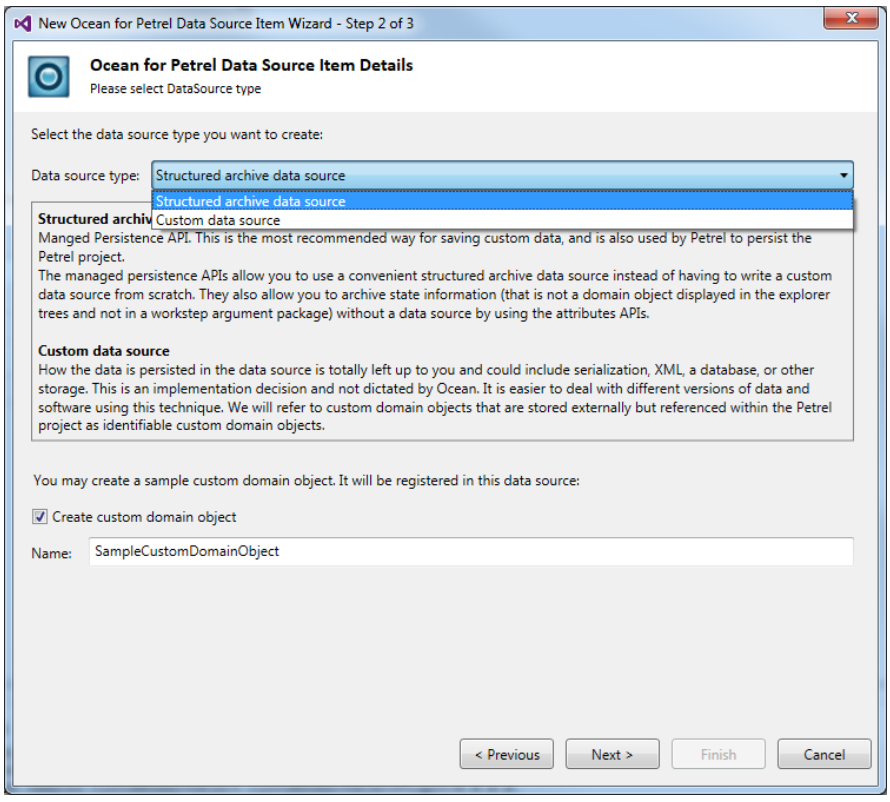


Figure 30: Adding structured archive data source

The following figure shows a preview of the Solution Explorer with the generated data structure archive data source class file and sample custom domain object class file.

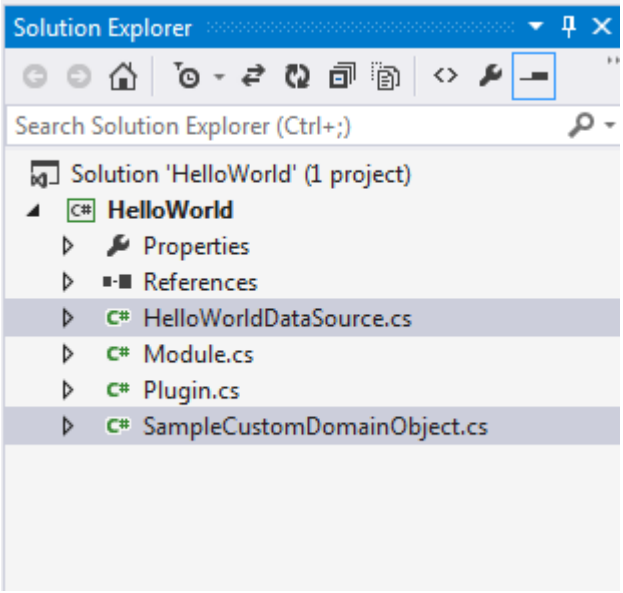


Figure 31: The Solution Explorer showing structure archive data source class file and custom domain object class file

To simulate a custom domain object that will save to Petrel project using structured archive data source, the simplest implementation would be to implement a command that will create a new instance of custom domain object to input pane.

```
public override void Execute(Slb.Ocean.Petrel.Contexts.Context context)
{
createDomainObject1_ = new SampleCustomDomainObject1("New CDO",DataSource1Factory.Get(DataManager.DataSourceManager));
```

```
PetrelProject.Inputs.Add(createDomainObject1_);
}
```

Figure 32: Simple command execute method to create sample custom domain object in input pane and will save to stuctured archive data source

Lastly, the generated structured archive data source framework code will output to message log pane as the custom domain object is being save to data source.

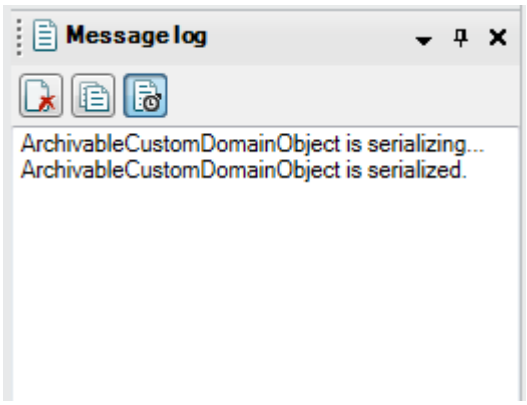


Figure 33: Output from OnSerialized event in custom domain object generated code

The second option only adds the framework to handle a data source. For additional details on adding data sources, refer to the *Ocean Developers Guide.*

The following figure shows a preview of the Solution Explorer with the generated data source class file.

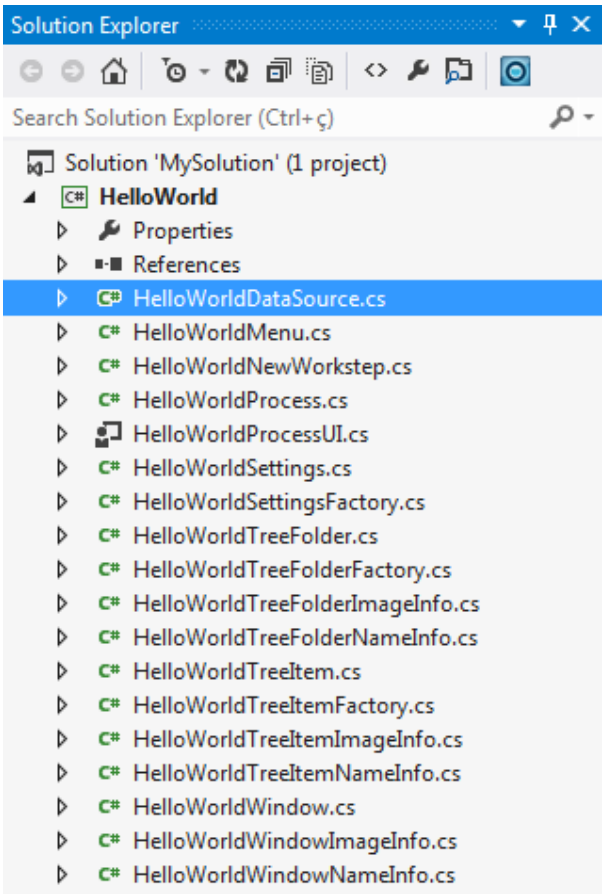


Figure 34: The Solution Explorer showing the New Data Source Class File

If you checked the **Register new classes in module** option in the Visual Studio Item dialog (Figure 12), then the Wizard adds the following code in the **Integrate** method of the selected module class.

```
public void Integrate()
{
    // Registrations:

    // TODO:  Add Module.Integrate implementation

PetrelSystem.AddDataSourceFactory(HelloWorldDataSourceFactory.
Instance);
}
```

Adding an Ocean Seismic Attribute

If you chose **Ocean Process** in the **Add/New Item** dialog (Figure 4), then the following screen opens:

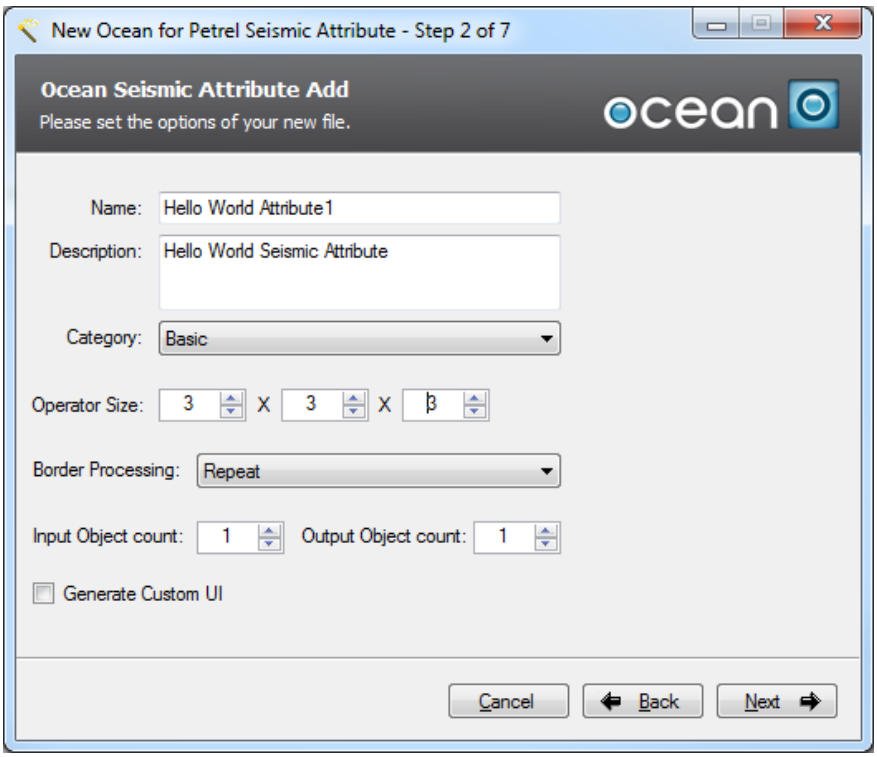


Figure 35: Ocean Seismic Attribute Wizard Page

1. Enter the appropriate information.

   Open the Geophysics / Volume attributes in Petrel to find the Petrel equivalents of the Ocean Seismic attributes. You can add your own attribute to that process's attributes. The name, description, and category of the attribute are self-explanatory.

The Ocean Seismic attribute represents an operator that completes calculations on a seismic cube. The operator size represents how many data samples the operator needs around the actual cell from the 3D data space. The attribute may have multiple input seismic cubes too.

2. Click **Next**.

The following figure shows a preview of the Solution Explorer with the generated seismic attribute class file:

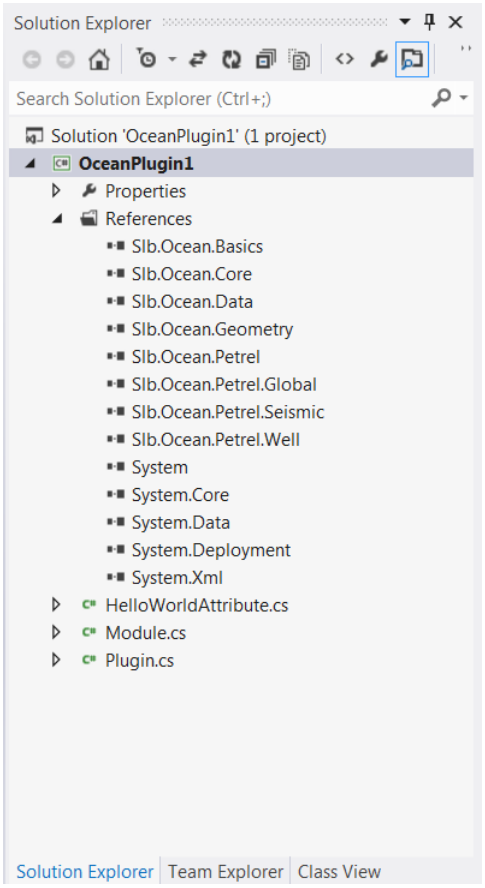


Figure 36: Custom Seismic Attribute in the Solution Explorer

The following source lines are added to the **Integrate** method of the module if the register option is enabled on the first page of the Wizard:

```
public void Integrate()
{
    // Registrations:

    // TODO:  Add Module.Integrate implementation
    if (PetrelSystem.SeismicAttributeService == null)
        throw new LifecycleException("Required
AttributeService is not available.");

PetrelSystem.SeismicAttributeService.AddSeismicAttribute(new
HelloWorldAttribute());
}
```

Adding an ECLIPSE Format Simulator Plug-in

This Ocean Wizard allows you to generate an **ECLIPSE Format Simulator plug-in** into your Ocean module. All files generated via this Wizard will be located under the **SimulatorPlugin** folder in your project.

1. Enter your plug-in simulator file name in the **Add New Item** dialog box (Figure 37). Note that if you use an existing file name, the old file will be overwritten by the new one. Click **Add** to proceed.

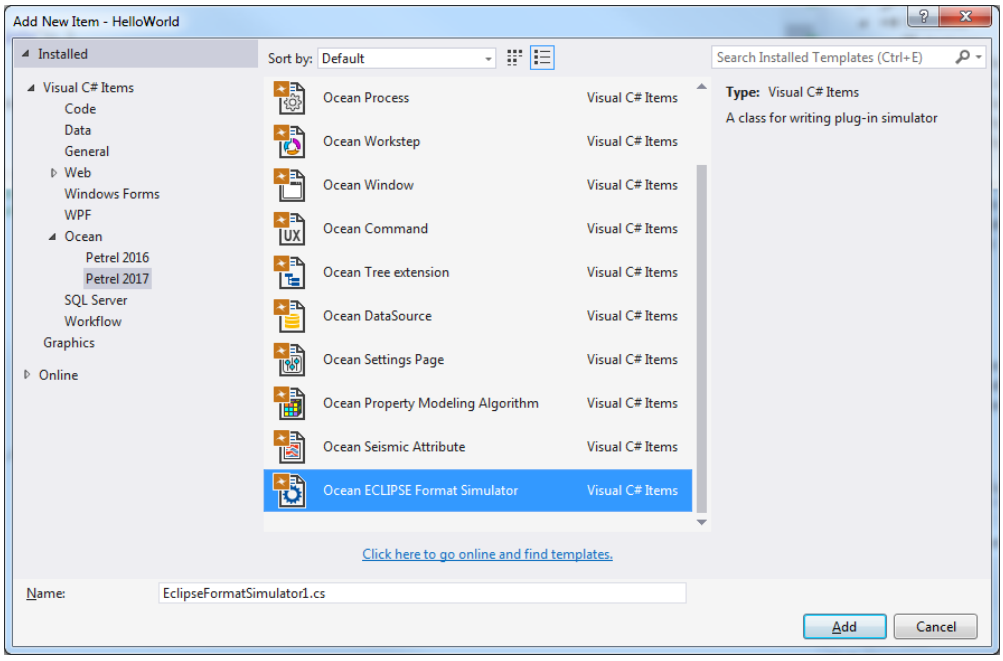


Figure 37: Ocean ECLIPSE Format Simulator template in Add New Item dialog box

2. In this step, you can decide whether to register your plug-in to the Ocean module or not.

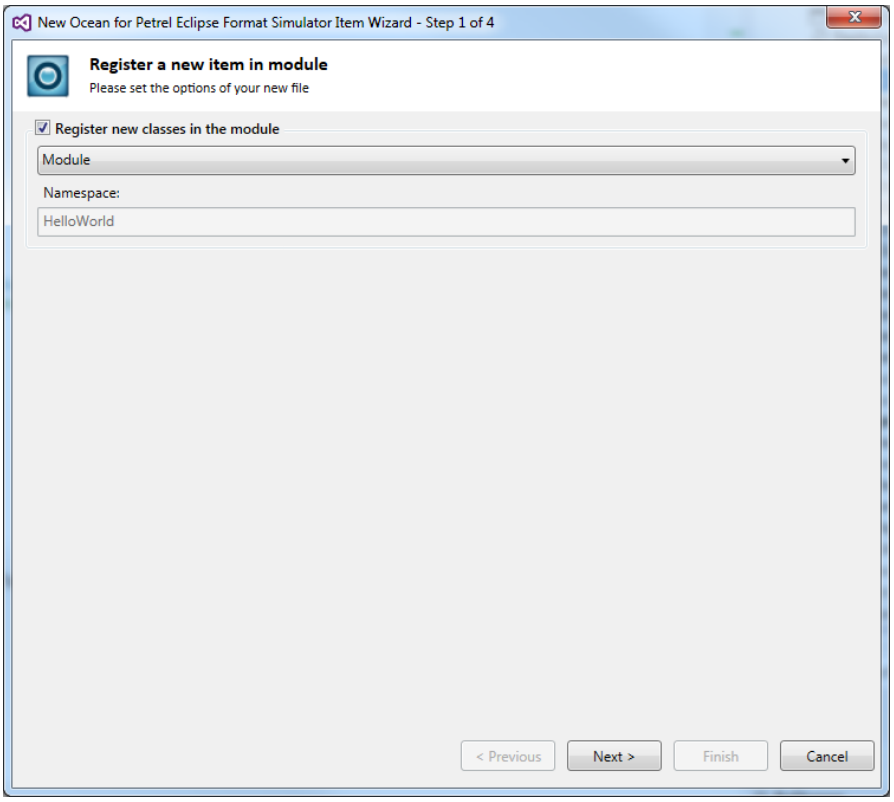


Figure 38: Ocean File Add dialog box

3. In this wizard page (Figure 39), you can choose the **ECLIPSE deck type** and whether you want to export keywords or not. Click **Next** to proceed or click **Cancel** to quit the wizard.

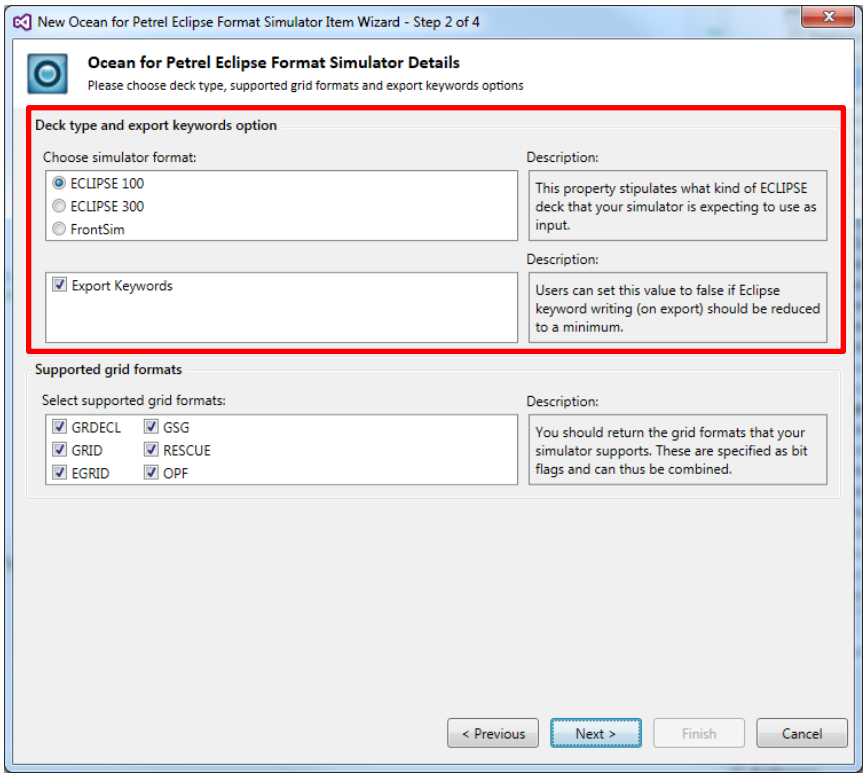


Figure 39: Deck Type and Export Keywords Option dialog box

4. All Grid Formats are supported by default. You can change the **supported grid formats** here (Figure 40).

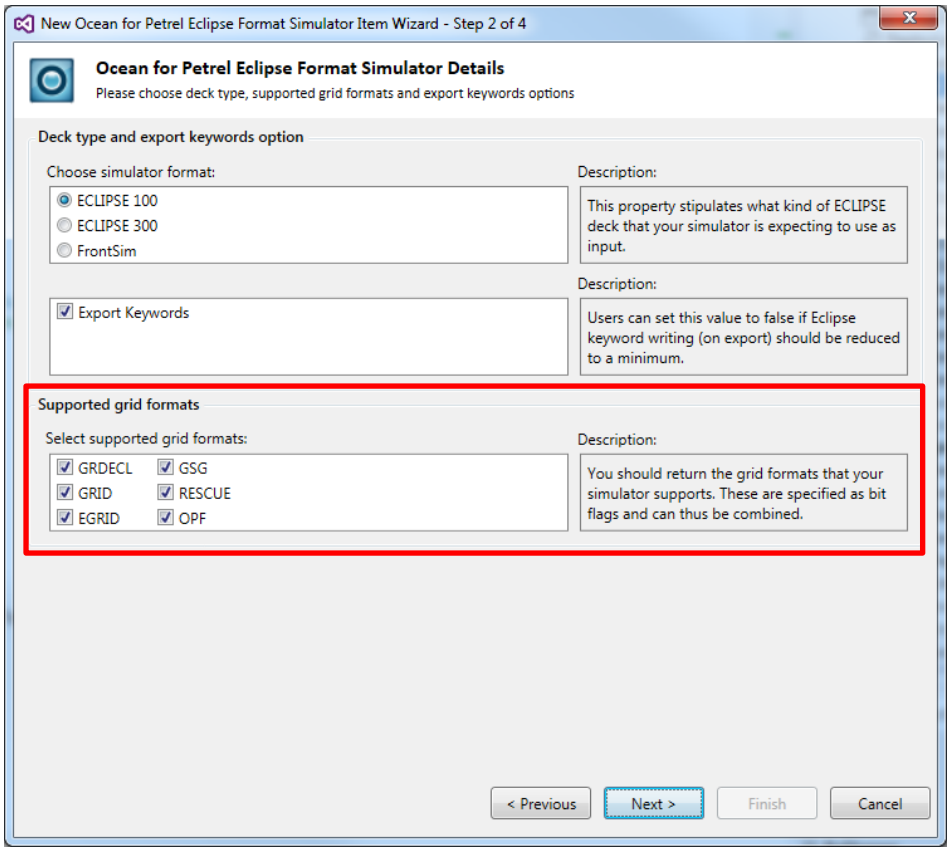


Figure 40: Supported Grid Formats dialog box

5. From the **Custom Tab Option** dialog box , you can set an option whether to create a custom tab or not. This custom tab will appear in the Petrel 'Define Simulation Case' dialog when your simulator is selected. You can input the class and tab names but the added custom tab will use a default icon image. This icon can be changed by overriding the ***IAppearance::Image*** property once the wizard has finished.

   If the check box is ticked then the following files will be generated (where Xxx is your class name) and added along with the EclipseFormatSimulator1.cs file:

   - XxxTabArguments.cs,
   - XxxTabControl.cs,
   - XxxTabLogic.cs
   - XxxTabUIFactory.cs

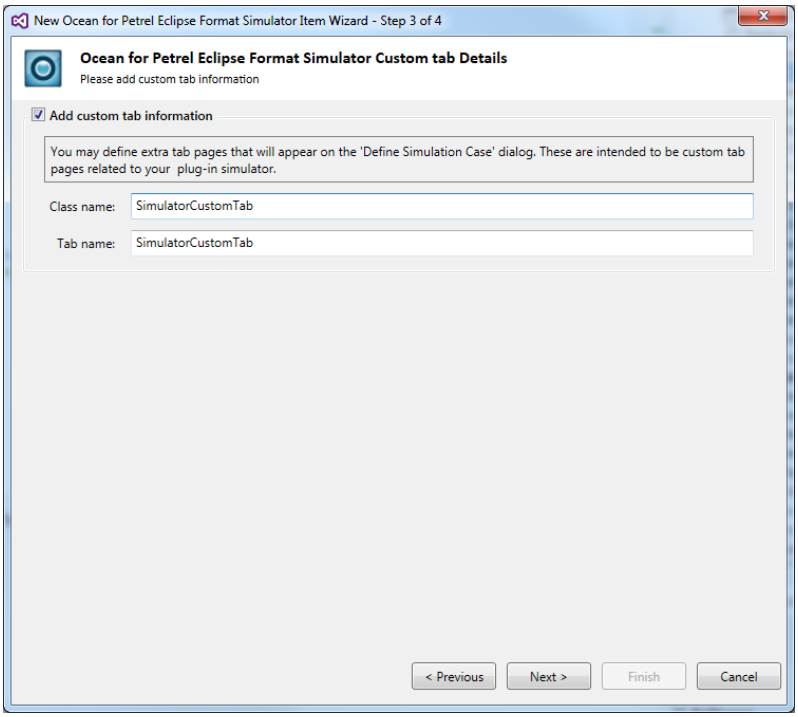


Figure 41: Custom Tab Option dialog box

6. The last step of the Wizard (Figure 42) shows a summary of the files which will be added to the project. In this example, only one file named EclipseFormatSimulator1.cs has been added (since the custom tab option was not ticked). If you tick the custom tab option, the appropriate files mentioned in step 4 will be also added to the **SimulatorPlugin** folder. You can also add additional references (assemblies) by ticking the **Additional Reference settings** check box.

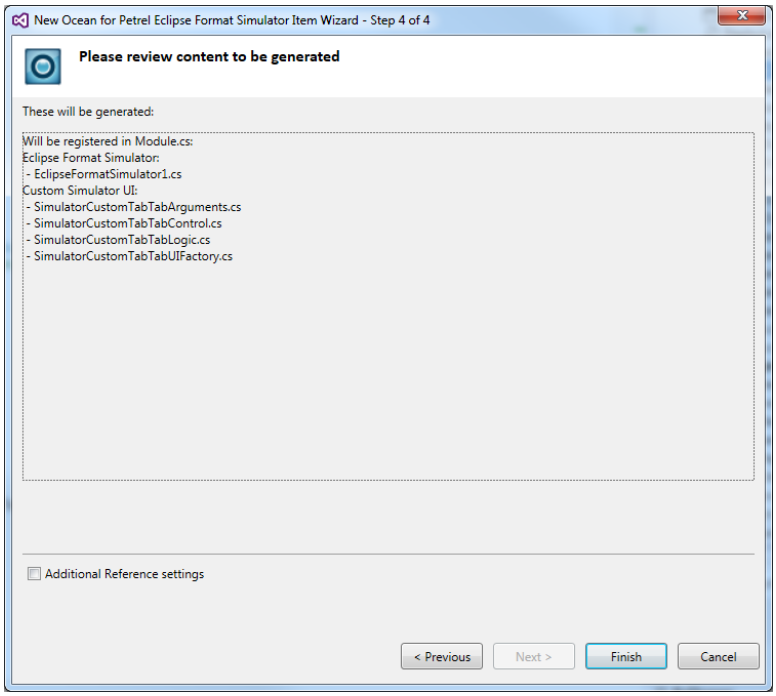


Figure 42: Completing settings dialog box

Adding a Plug-in project

More than one plug-in project can be added to the same solution. The plug-in project allows you to generate a basic plug-in implementation with additional project settings to help developing the plug-in. Basically, a plug-in groups multiple modules together, and lets you set up dependency between plug-ins.

There are some rules and recommendations using the plug-ins. These are checked by Wizard:

- In a solution there should be only one plug-in. If there are 2 or more, then they should contain only different modules. One module must belong to only one plug-in. The wizard warns you when you try to create a new plug-in when there is another one already:

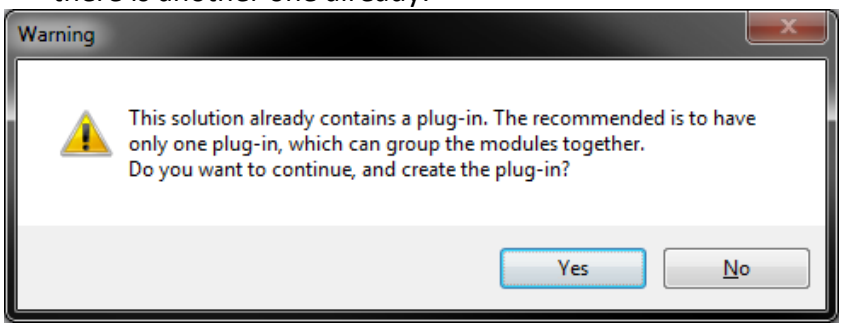


Figure 43: Warning message. Another plug-in is detected in the solution.

- In one project there has to be one (and only one) plug-in. This is a very strict rule; do not create more than one plug-in class into a project. The wizard lets you do it, but we warn you, that the registration of such a plug-in will fail; you will not be able to use any of those plug-ins.

Generating a plug-in project or file:

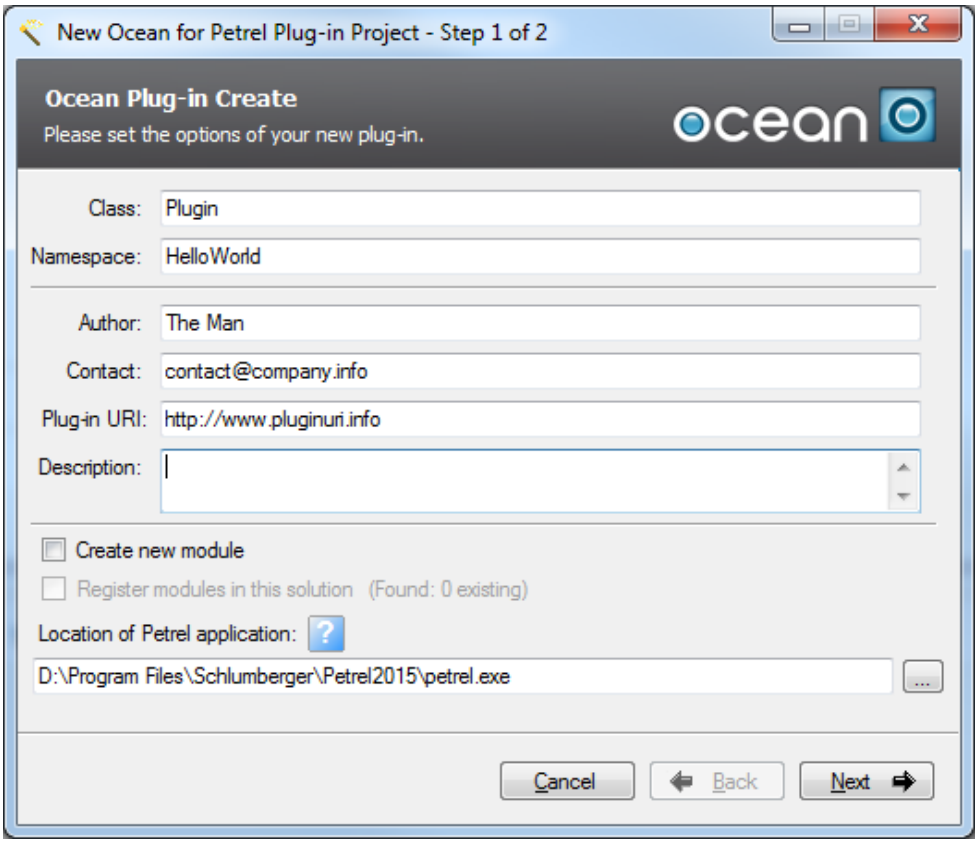


Figure 44: Plug-in wizard, 1st page.

In this page, you can specify the plug-in metadata information such as the author, contact information and a description. This information is redundant if you use the old IModuleAppearance on your modules, but the information given in the plug-in will be the main source.

If you check the Register existing modules in the solution checkbox, then you can select which modules want to register from the detected module list:

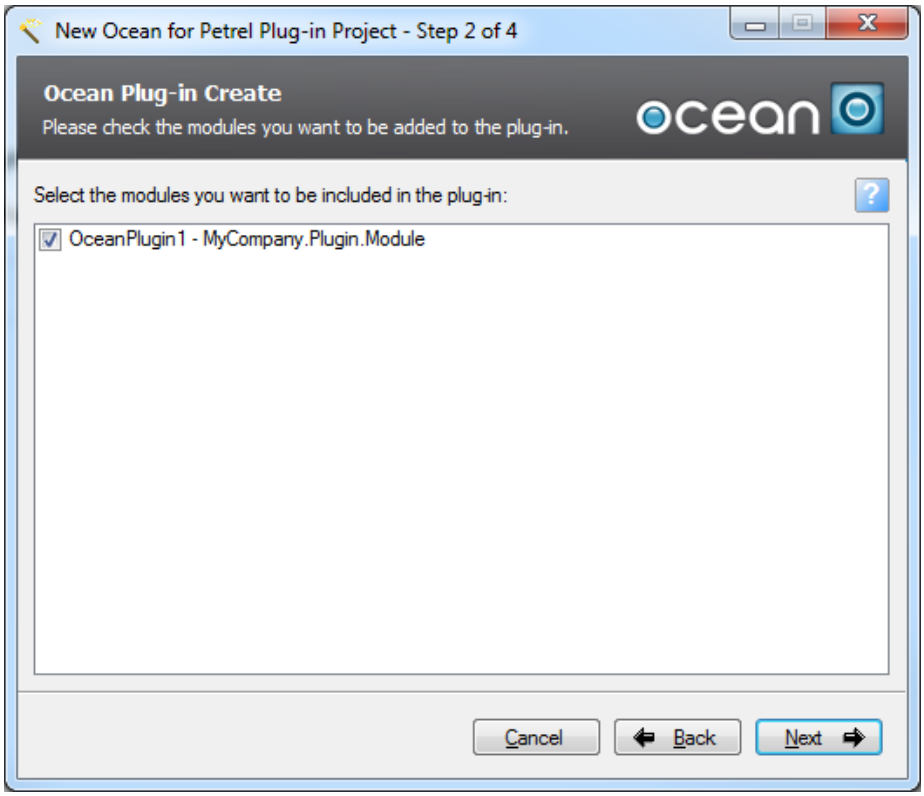


Figure 45: Plug-in wizard, 2nd page.

In order to use the plug-in, there must be at least one module registered in it.

You also have the option to generate a new module together with the new plug-in. If you choose to generate a new module, you will get the same module-generator wizard pages you are already familiar with.

During the building of your plug-in, as a post build event, the plug-in gets registered in Petrel, so you need to build your plug-in at least once, and after that, the plug-in is usable from Petrel.

Adding a New Property Modeling Algorithm

If you chose **Ocean Property Modeling algorithm** in the **Add / New Item** dialog then you will see the following Wizard page.

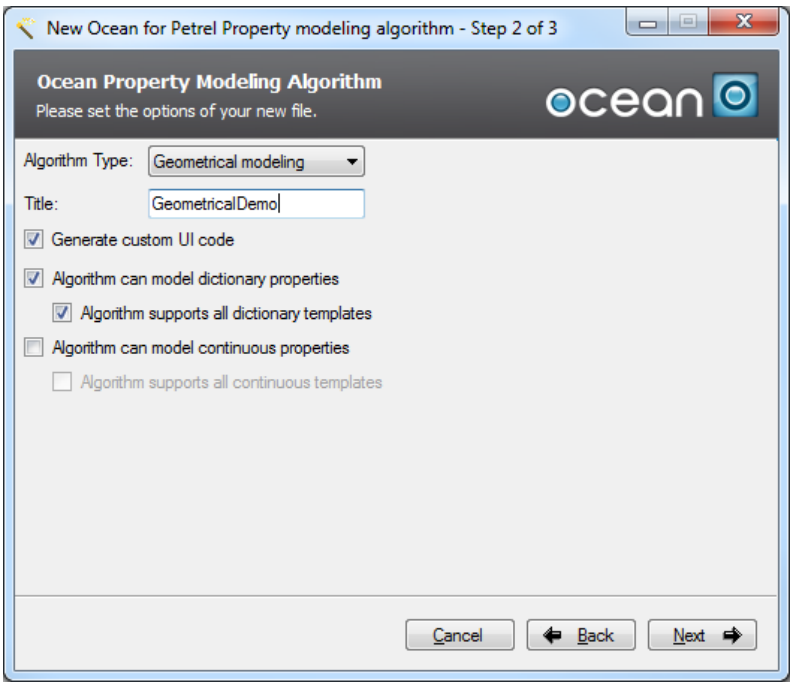


Figure 46: Second page of Property Modeling wizard

Geometrical modeling algorithm

For adding a new geometrical modeling algorithm select “Geometrical modeling” algorithm type (Figure 46). Specify the following options:

- Title: Enter the title of the algorithm. This text will be displayed on the Geometrical modeling process dialog.

- Generate custom UI: If selected wizard will generate an empty UserControl and the necessary UI registration code
- 'Algorithm can model dictionary properties': If selected generated algorithm will be present in Geometrical modeling if a dictionary property is selected.
- 'Algorithm supports all dictionary templates': Generated code will return all dictionary templates, otherwise SupportedDictionaryTemplates method needs to be filled with supported templates.
- 'Algorithm can model continuous properties': If selected generated algorithm will be present in Geometrical modeling if a continuous property is selected.
- 'Algorithm supports all continuous templates': Generated code will return all continuous templates, otherwise SupportedTemplates method needs to be filled with supported templates.

After building the project and starting Petrel start geometrical modeling process. You will see the algorithm in the geometrical modeling UI (Figure 47). It can be executed but will do nothing. //TODO sections in the code must be filled to get a meaningful algorithm.

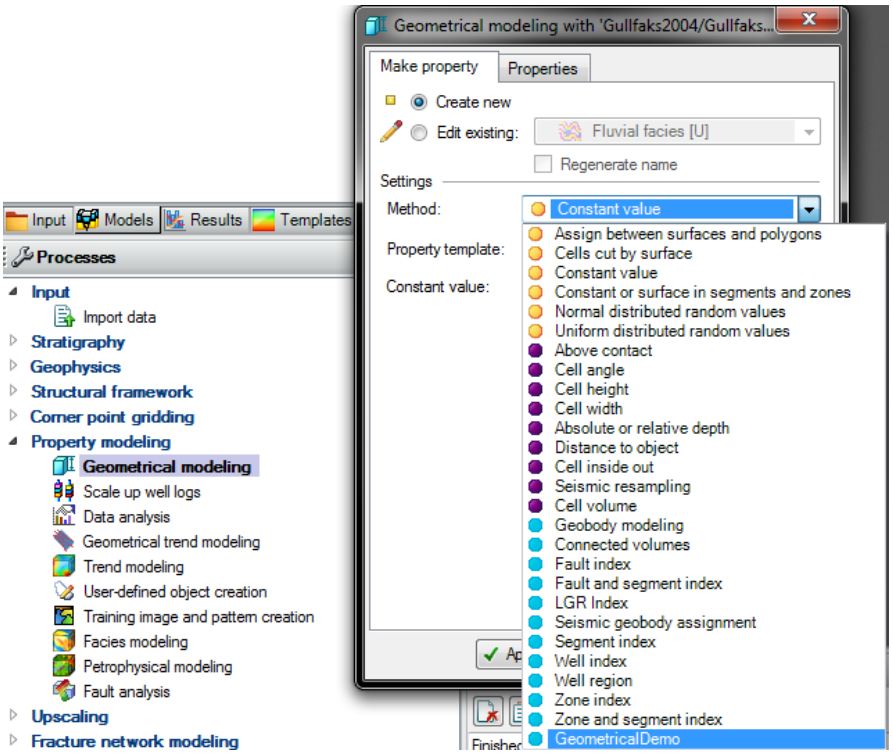


Figure 47: Geometrical modeling algorithm in Petrel

Facies modeling algorithm

For adding a new facies modeling algorithm select “Facies modeling” algorithm type. (Figure 48)

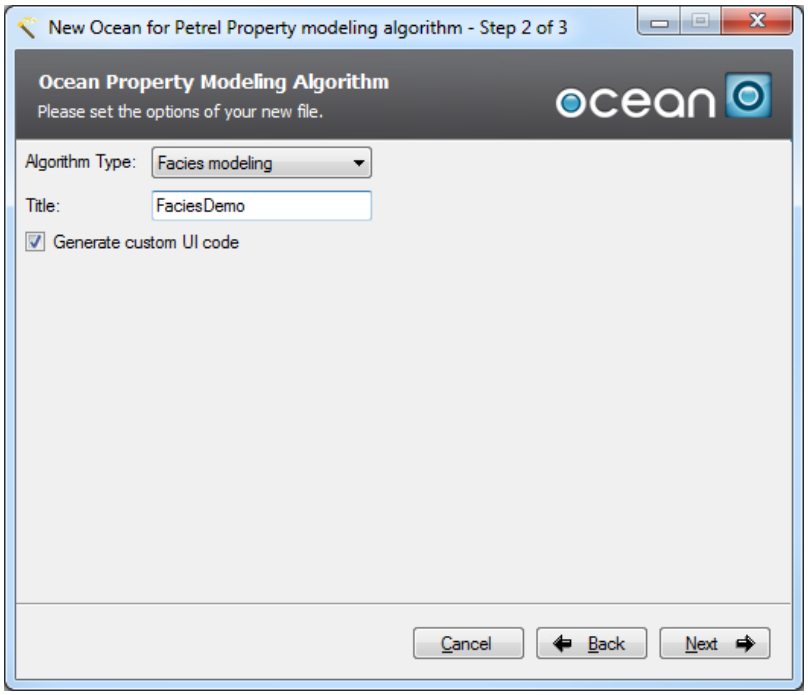


Figure 48: Facies modeling algorithm

Specify the following options:

- Title: Enter the title of the algorithm. This text will be displayed on the Facies modeling process dialog.
- Generate custom UI: If selected wizard will generate an empty UserControl and the necessary UI registration code

After building the project and starting Petrel start facies modeling process. You will see the algorithm in the facies modeling UI (Figure 49). It can be executed but will do nothing. //TODO sections in the code must be filled to get a meaningful algorithm.

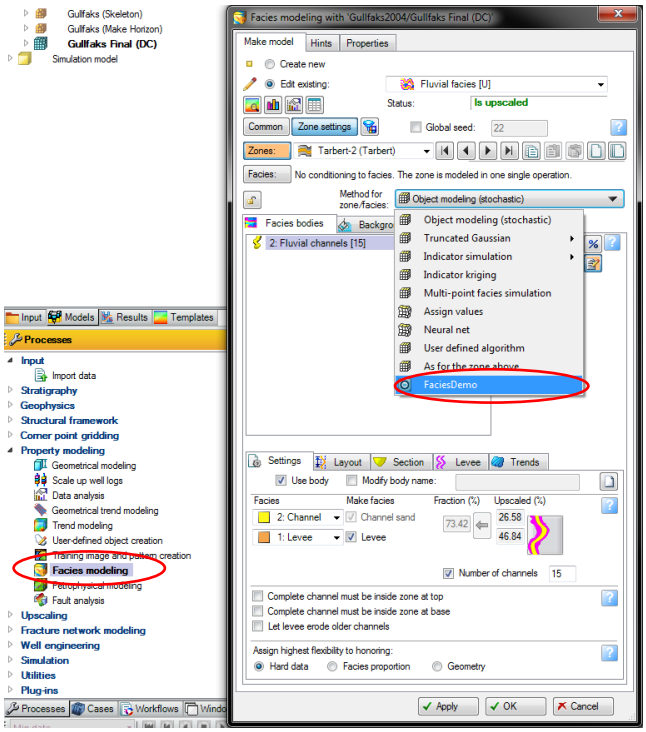


Figure 49: Facies modeling algorithm in Petrel

Petrophysical modeling algorithm

Identical to facies modeling.

Adding a PIP Project

This project type generates an installation package, what the Petrel application can consume. Instead of using the operating system to install and manage extensions, let Petrel do the work, and manage its extensions by itself. This is the way to deploy modules and plug-ins. However, the PIP package cannot cover special use cases such as pre/post installation steps, custom actions, installation of 3rd party prerequisites, etc. In case your plug-in package needs such installation steps, please see the Ocean Plug-in Installer project.

Prerequisite: the solution has to contain at least one plug-in project (project with a plug-in class).

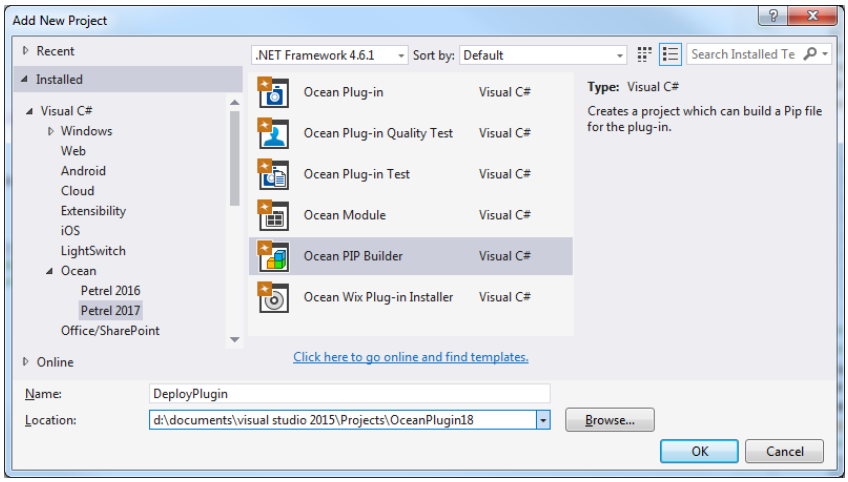


Figure 50: PipBuilder project

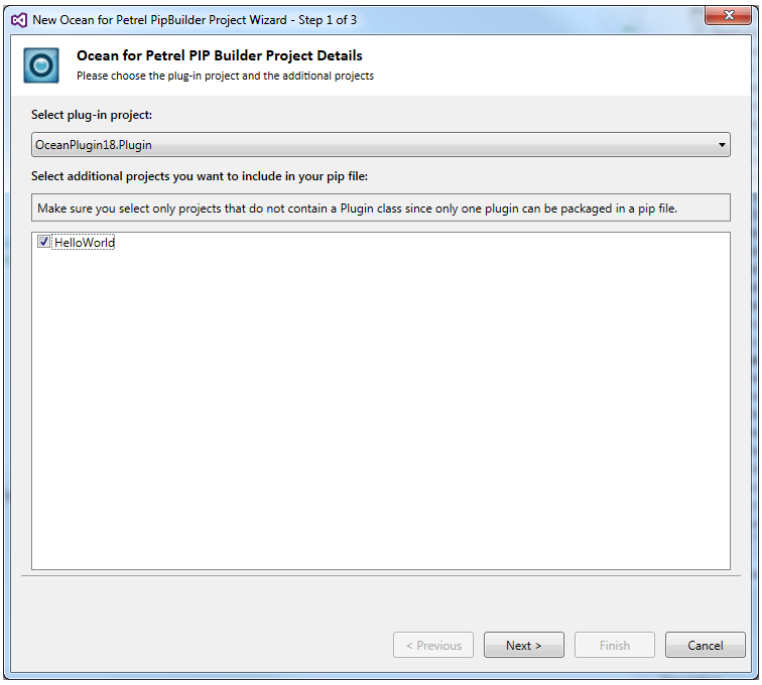


Figure 51: PipBuilder project wizard

In the wizard page, you can specify the main plug-in for this PIP file. The PIP file must have one and only one plug-in in it, this plug-in will be detected by the PluginPackager, and it will be registered into the PluginManager during the installation of the PIP.
In the checklistbox, you can add additional projects from the actual solution. The output of the selected projects will be added to the PIP file together with the plug-in container assembly.

Note: currently the wizard does not support dependency detection for the plug-in assembly. It means, that in case the plug-in assembly (or any other assembly selected in the checklistbox) has a dependency to another project, then it will not be added to the PIP file automatically. You have to add all of the required dependencies.

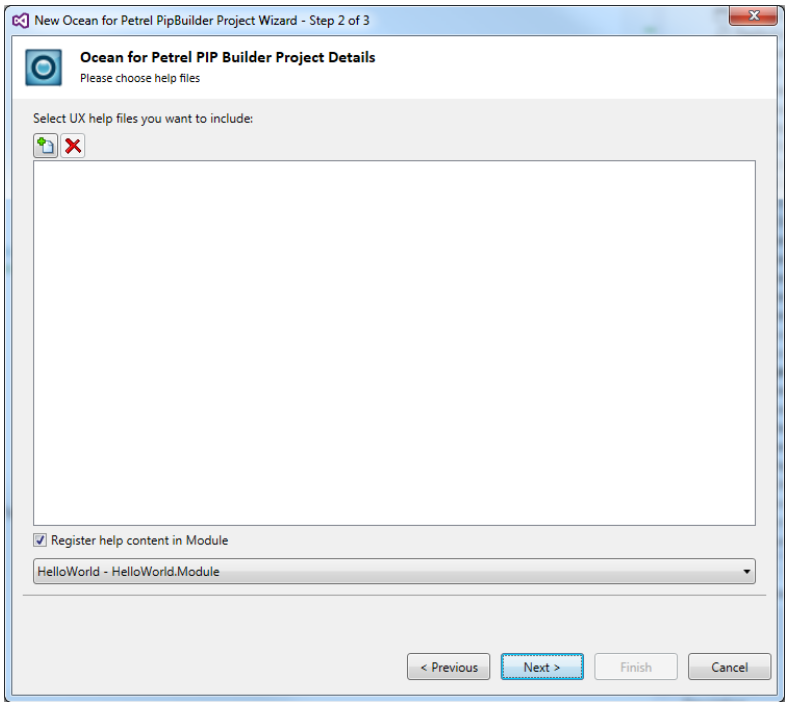


Figure 52: PipBuilder project wizard

In step 2 of the wizard page, you can add help file for classic and UX mode for your plug-in. You can drag and drop help files from Windows Explorer into this wizard.

Editing the DeployList.xml

The DeployList.xml describes the internal structure of the PIP file. This XML is loaded during the build of the project, and the DeployCopier command line tool (placed to the project directory by the Wizard) will copy all the files and project outputs registered in this xml to a temporary folder. It also keeps their folder hierarchy settings. From this temporary folder, the PluginPackager tool (deployed by the Petrel installer to the petrel root folder) will pack them to the target folder of the PIP project. (By default: bin\debug and bin\release)

The temporary folder can be changed by editing the Pre and PostBuild events of this project, and changing the paths containing the “copytemp” folder.

Deploylist.xml can contain files and project outputs

The files are normal files, they can be resources (pictures, sounds, videos, databases, etc). They will not be processed, only copied and packed.

The project outputs are project files, preferably from the same solution as the PIP builder project itself. The project files will be processed by the DeployCopier tool, the output path and compiled binary file name will be retrieved using the actual platform and configuration settings. At the end the project output will be copied to the temporary folder.

The OceanSDK installs a new context menu item into Visual Studio. It can be reached by right clicking on the DeployList.xml file in the Solution Explorer:

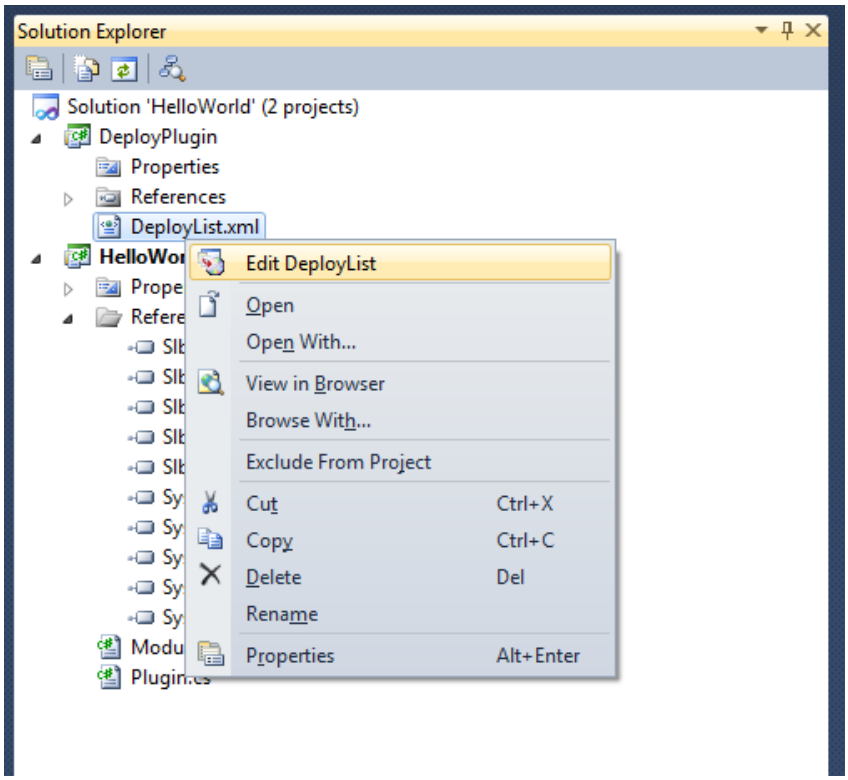


Figure 53: Edit DeployList context menu item

Clicking on this menu item, a small editor appears:

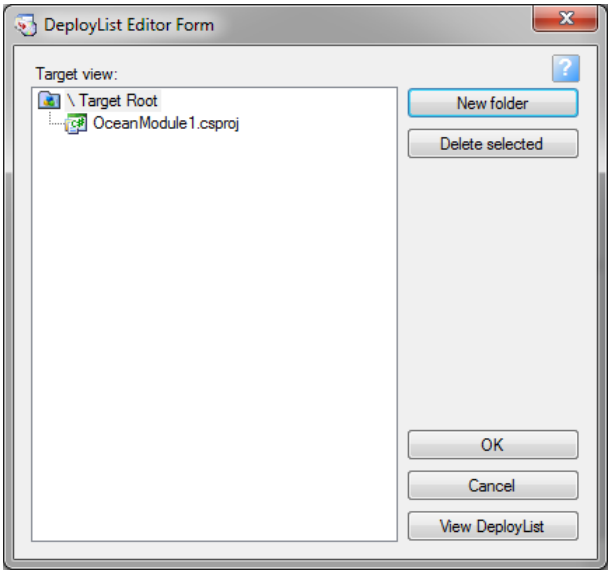


Figure 54: DeployList editor dialog

In the tree view on the left side contains the folders and files of the Pip file in the same hierarchy it will be installed on the target machine. Use drag and drop from either the Solution Explorer, or Windows Explorer to add new files to it.

If you want to add a project output from the existing solution, just drag that project from the Solution Explorer. The editor detects if the dragged item is a project, and it automatically adds Project dependencies to that project, so building the Pip file will build that project also.

You can rename the files and folders with F2 key or with clicking on the item again. These settings will be applied to the target name; it will not change the actual name of the file.

Adding Ocean Wix Plug-in Installer Project

This project can be used when a simple PIP file installation with PluginManager does not meet the needs. The result of this project template is an MSI installer project that can be further customized by adding extra installer actions. It is also possible to combine more than one plug-ins (PIP files) into one MSI installer as Plug-in bundle. Please note however plug-ins installed together as bundle cannot be uninstalled individually.

We recommend using the Ocean Wix Plug-in Installer in case you have a special need which the PIP project cannot handle, e.g.: custom actions, pre/post installation steps, installation of 3rd party prerequisites, etc.

The recommended way of using this Ocean Wix Plug-in Installer template is to first add one or more Ocean Plug-in project(s) to the solution then add a PIP Builder project for each plug-in, compile the PIP builder project(s) then add this Ocean Plug-in Installer project for the PIP file(s).

To add a new Ocean Wix Plug-in Installer to the project:

1. Once you have Ocean Plug-in and PIP Builder project in the solution, compile the PIP builder project and make the selections shown in the following figure.

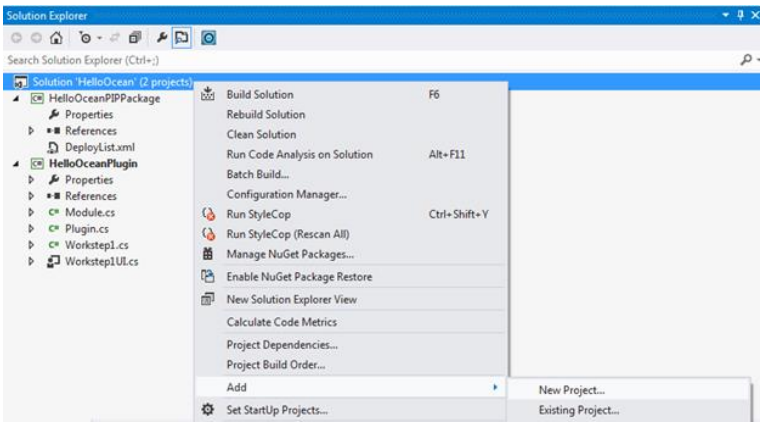


Figure 55: Visual Studio Context Menu to Add a New Project

2. Choose the Ocean Wix Plug-in Installer on the displayed project dialog. Remember to set the Project Name. (See Figure 4.)
3. Click **Next**.

The Step 2 window opens:

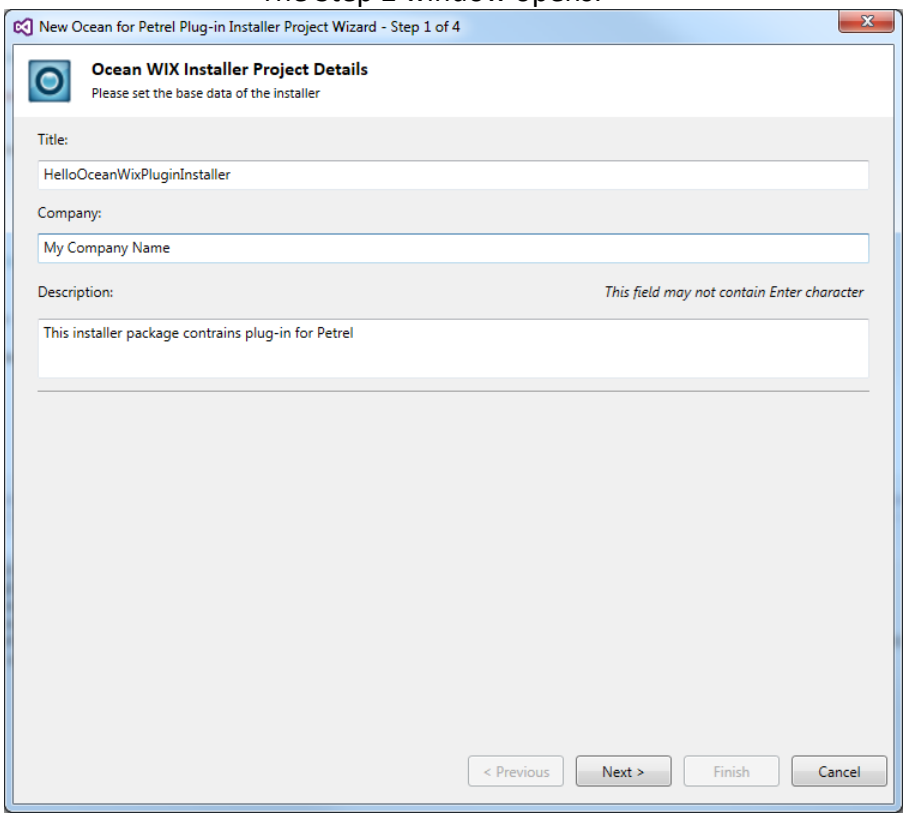


Figure 56: Step 2, Setting Title, Company and Description

4. Enter the appropriate information as follows:

- **Title**: The text entered here is displayed on the title bar of the installation kit. In this example, enter "HelloOceanWixPluginInstaller". This will be displayed in the installed programs list at Control Panel -> Add Remove Programs.
- **Company**: The text typed here is added to the properties of the MSI kit. This information can viewed in the summary tab under the properties of install kit.
- **Description**: The text typed here is added to the properties of the MSI kit. This information can be viewed in the comments section under the properties of the install kit.

5. Click **Next.**

   The Wizard opens the Step 3 window as shown in the following figure.

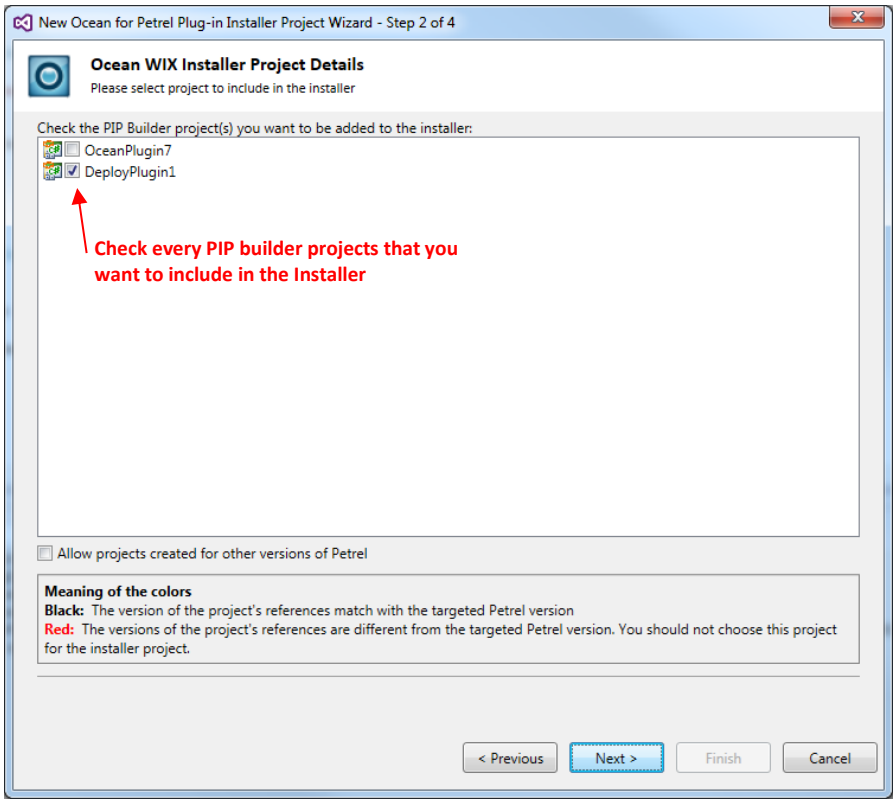


Figure 57: Step 3, Setting the Included PIP Builder projects

This dialog lists all of the projects available under the current solution. This example includes the "HelloOceanPlugin" Ocean Plug-in project and the "HelloOceanPIPPackage" PIP Builder project. This Installer project needs the PIP files (the outputs) of PIP Builder projects. Select those you want to be included in the installer MSI file. You may add other PIP files manually (not located in this solution) later.

6. Click **Next**.

The Wizard opens the Step 4 window as shown in the following figure.

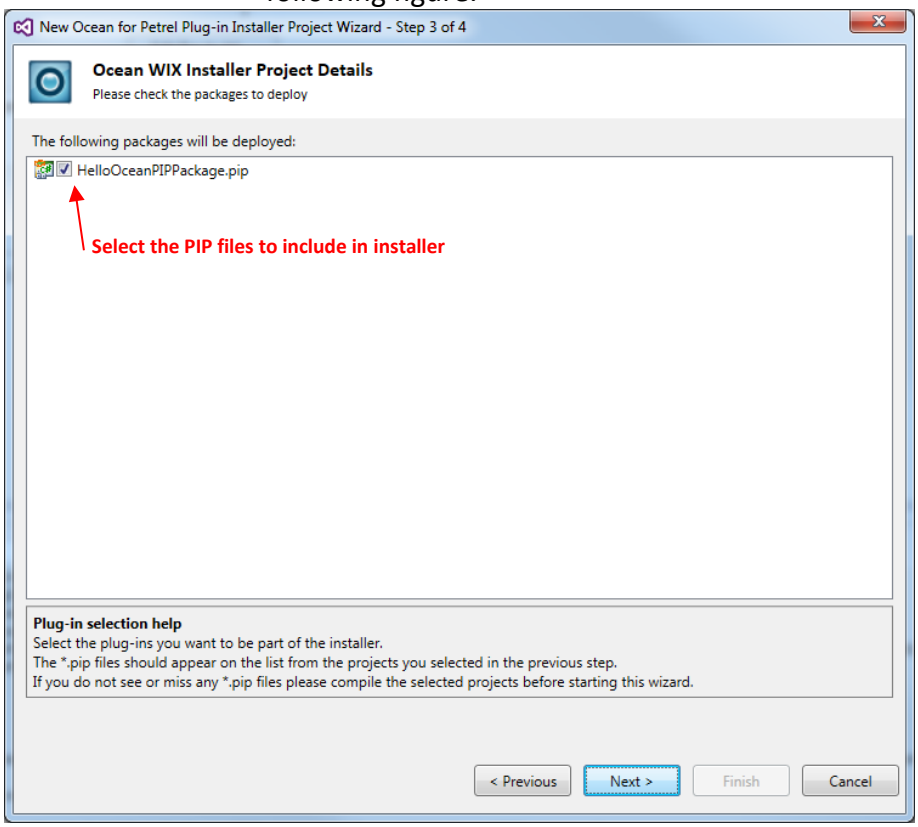


Figure 58: Step 4, Selecting PIP files

The PIP files from the PIP builder projects selected in the previous step are listed here. If you selected the PIP Builder project at previous step but the PIP file is still not listed here, please compile the PIP builder project (the “HelloOceanPIPPackage” project in this example) and start the wizard again. Only existing PIP files can be added to the installer project.

You can decide whether the plug-in should be installed for the current user (eventually running the installer) or for all users on the target computer.

7. Click **Next**.

   The Wizard displays the summary page with the information what will be installed.
   In this example it is:
   “Plug-ins to install:
   HelloOceanPIPPackage.pip” – that contains the HelloOceanPlugin

8. Click **Finish**.

   Note: From 2015.1 onwards, wizard will automatically update all version number to ensure clean uninstall in scenario where plugin is being update.

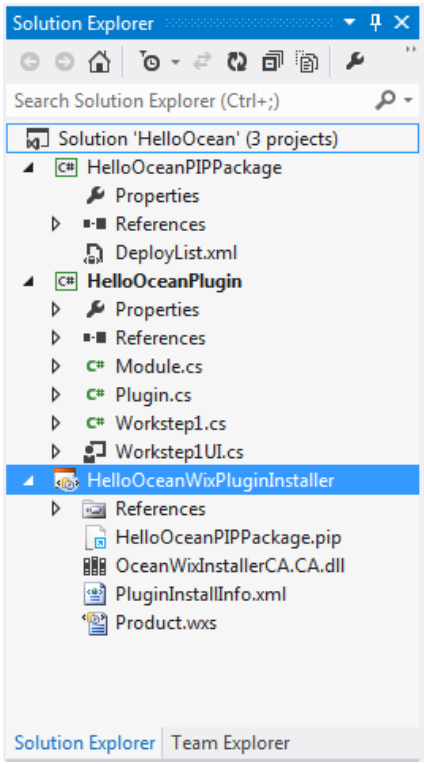


Figure 59: Preview of Installer Files in Solution Explorer

The “HelloOceanWixPluginInstaller” project has been created and ready to compile. Its output is a HelloOceanWixPluginInstaller.msi that can be shipped to the users (or upload to Ocean Store). Please check the ReadMe.txt file added the project for more options. There is no “Primary project output” in the installer project instead, the PIP files are added to the project that you selected in the wizard. MSI installer will call PluginManager to install these PIP files. There is one more file worth mentioning, the PluginInstallInfo.xml.

Customizing the PluginInstallInfo.xml

This file contains the information about the plug-ins necessary for PluginManager when installing or uninstalling the plug-ins.

To edit this file, please right click on it and choose the Open… menu item. Double-clicking on it will not do the same, it will show its file system settings instead.

The structure of this XML is:

```
<?xml version="1.0"?>
<PluginInstallInfo
      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Plugins>
    <Plugin Id="://HelloOcean.Plugin/1.0.0.0">
      <Name>Plugin</Name>
      <PipFilePath>
        {Project_Folder}\bin\Debug\HelloOceanPIPPackage.pip
      </PipFilePath>
    </Plugin>
  </Plugins>
</PluginInstallInfo>
```

Please note the followings:

- The < PluginInstallInfo> node is an XML root; there must be only one in the XML document. The XML must be a valid, well-formed XML.
- There can be more than one <Plugin> nodes in the XML, all of them will be used during the installation/uninstallation.
- The PIP files for <Plugin> nodes added manually to the XML should be part of the installer project and listed in Application Folder in order to be in the MSI file.
- Files can be renamed but PIP file content cannot be changed.
- If you have more than one Plug-ins in PluginInstallInfo.xml file (and corresponding PIP files in the project) they are regarded as one bundle and handled as such. That plug-ins are installed/uninstalled together only.

The Plug-in Manager (that eventually installs the plug-ins) will install what is in the pip files and ignores all other files deployed by MSI. If you need something extra to what PluginManager does then you might need to add a Custom Installer step.

Creating an Ocean Plug-in Test Project

The Plug-in Test Project is used for unit testing an existing Ocean Plug-in through the Ocean Testing Framework. Before starting to write the unit tests, Ocean Testing Framework demands several configuration steps in order to create a test project, as follows:

- The test project has to contain a reference to the assembly of the plug-in to be tested;
- The test project assembly has to be named with the same assembly name of the plug-in to be tested plus the suffix **.Test**. Example: if the assembly name of the plug-in to be tested is MyCompany.MyPlugin, then the test assembly name must be MyCompany.MyPlugin.Test;
- The test project output path (destination where assembly and compilation artifacts are stored) must be set to Petrel installation folder;
- Reference paths must be pointed to Petrel installation and Public (PETREL_HOME/Public) folders as well as to any additional required folder;
- A reference to the Ocean Testing Framework assembly (Slb.Ocean.Petrel.Testing.dll) must be added to the test project;
- The Petrel configuration file (petrel.exe.config) must be copied into test project, renamed with the same test project assembly name (Ex.: MyCompany.MyPlugin.Test.dll.config), and have its property *Copy to Output Directory* to *Copy always*;
- Add references to assemblies of the used unit testing framework (NUnit).

Note: Due Petrel implementation restrictions, the Ocean Testing Framework supports only NUnit as test framework.

There is clearly a big time consuming and manual effort only for configuring a test project. To overcome such overhead and to boost the developer productivity, the Ocean Plug-in Test Project wizard wraps up all of the above steps in a few clicks. In a solution containing the plug-in project to be tested:

1. Click on File > New… > Project…;
2. Under Ocean templates, select Ocean Plug-in Test, provide a project name and mark the option Add to solution;

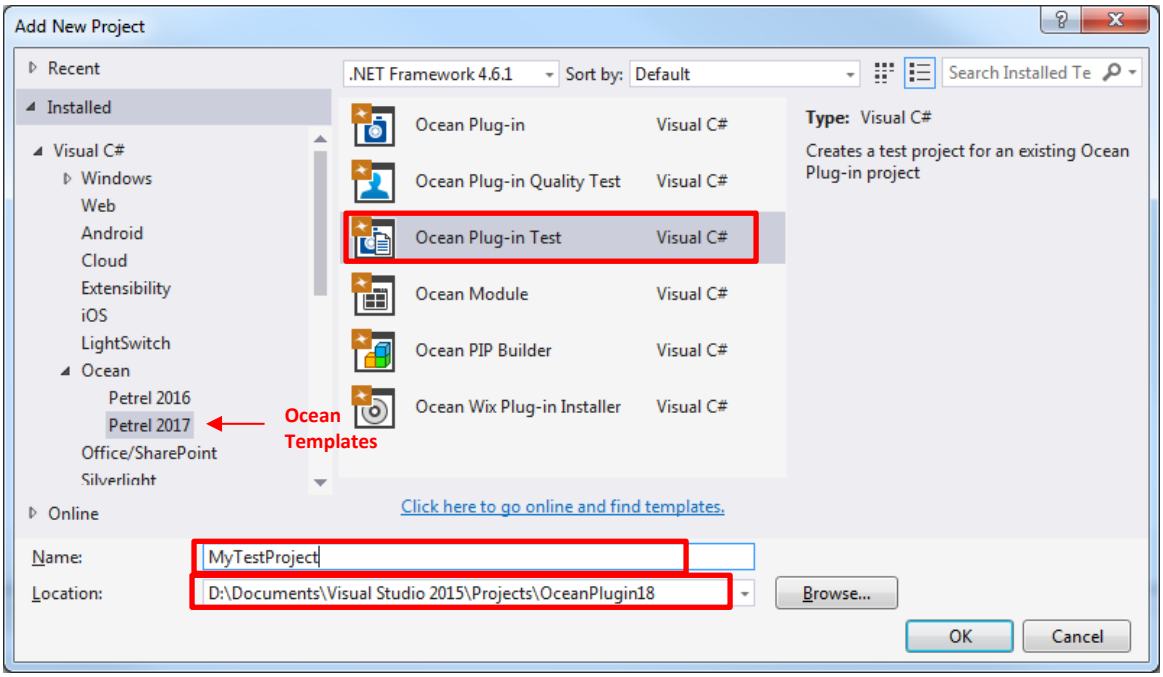


Figure 60: New Plug-in Test Project wizard launch

Add the project to the current solution

3. Click on OK button;
4. In the next screen, select which project will be tested as well as the NUnit installation folder. Note that only Ocean Plug-in projects are listed in the combo box;

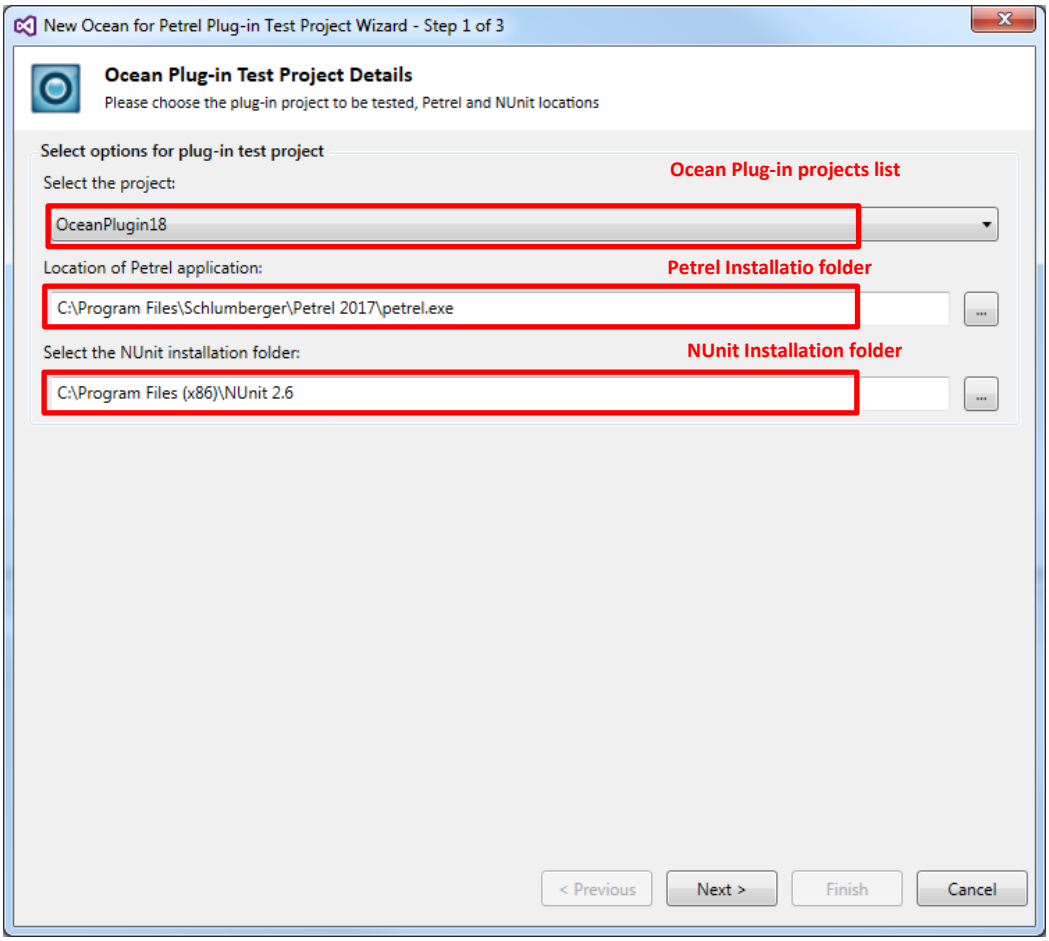


Figure 61: Plug-in Test Project configuration

5. Click on Next button;
6. In the next screen, review the test project settings and click on Finish button.

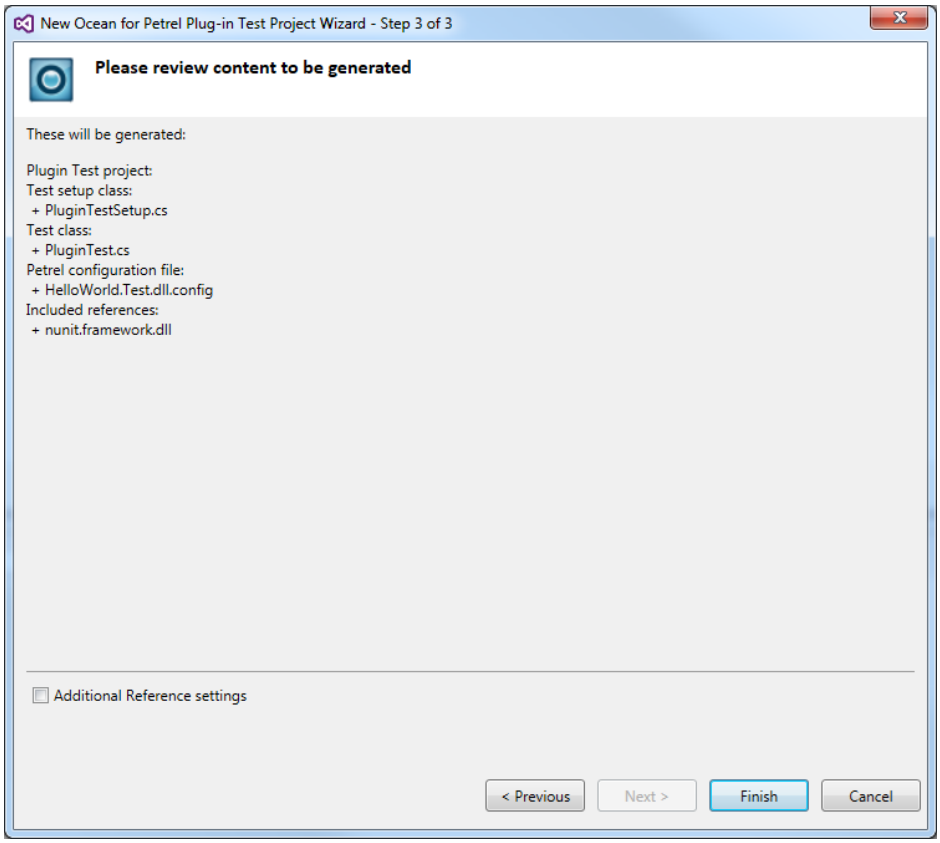


Figure 62: Plug-in Test Project settings review page

7. A new Plug-in Test Project is created, including all of its required configuration, a test setup class and a test template class.

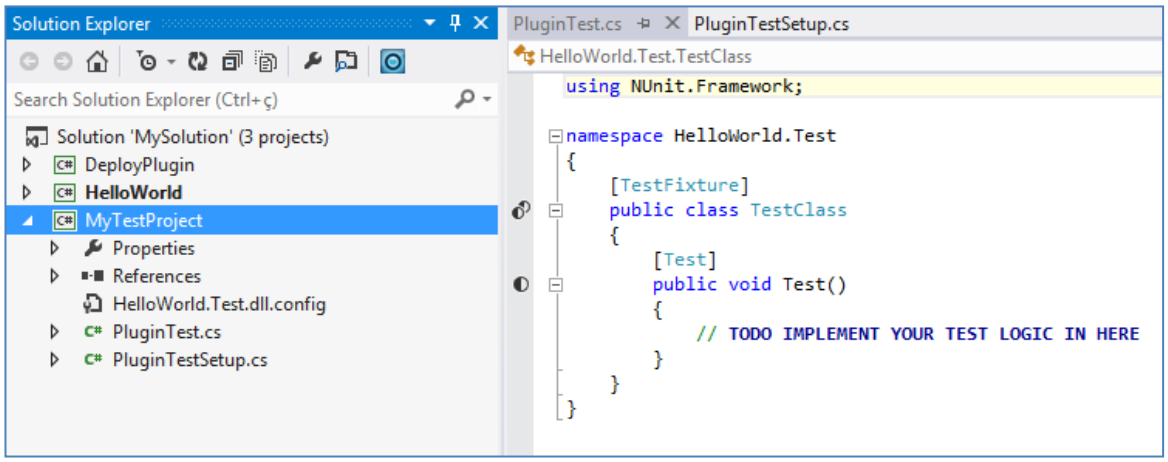


Figure 63: Plug-in Test Project review settings page

8. After the project files have been created, the developer can start coding his unit test against the plug-in to be tested.

THE OCEAN DEVELOPERS TOOLS

Ocean provides a set of tools in Visual Studio to simplify the quality control, the testing process, and the upgrade of Ocean Plug-ins. This chapter provides an overview of the Ocean Developers Tools in Visual Studio features.

Ocean Quality Assistant

Ocean Quality Assistant is a Visual Studio extension that offers 3 main features,

- Run Automated Acceptance Tests;
- Manage Manual Acceptance Tests;
- Run the fixed set of Sanity Unit Tests.

Acceptance rules are updated for every release as per rules from Ocean Store acceptance criteria. Updated version of the rules is available from Ocean Portal (click on link from Ocean Quality Assistant to Ocean Portal to download latest acceptance criteria check list)

Ocean Quality Assistant will be deployed by OceanSDK installer.

Accessing Ocean Quality Assistant

In Visual Studio, Ocean Quality Assistant can be accessed through,

- VS main menu > Ocean > Ocean Quality Assistant
- Solution Explorer > 

Note: It is not mandatory to have an opened VS solution to access Ocean Quality Assistant.

Note: It is not mandatory to have an opened VS solution to access Ocean Quality Assistant.

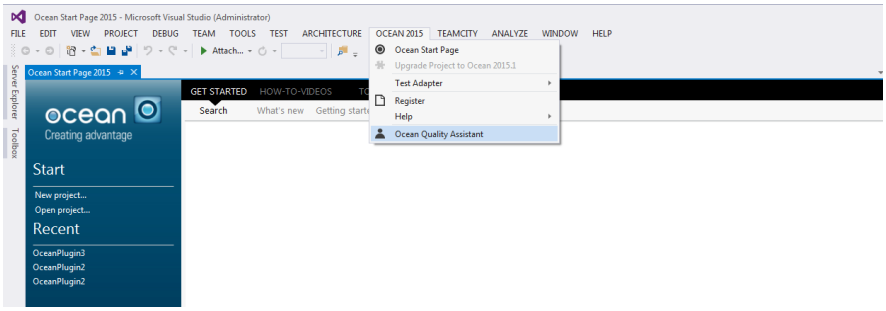


Figure 64: Access Ocean Quality Assistant from the main menu

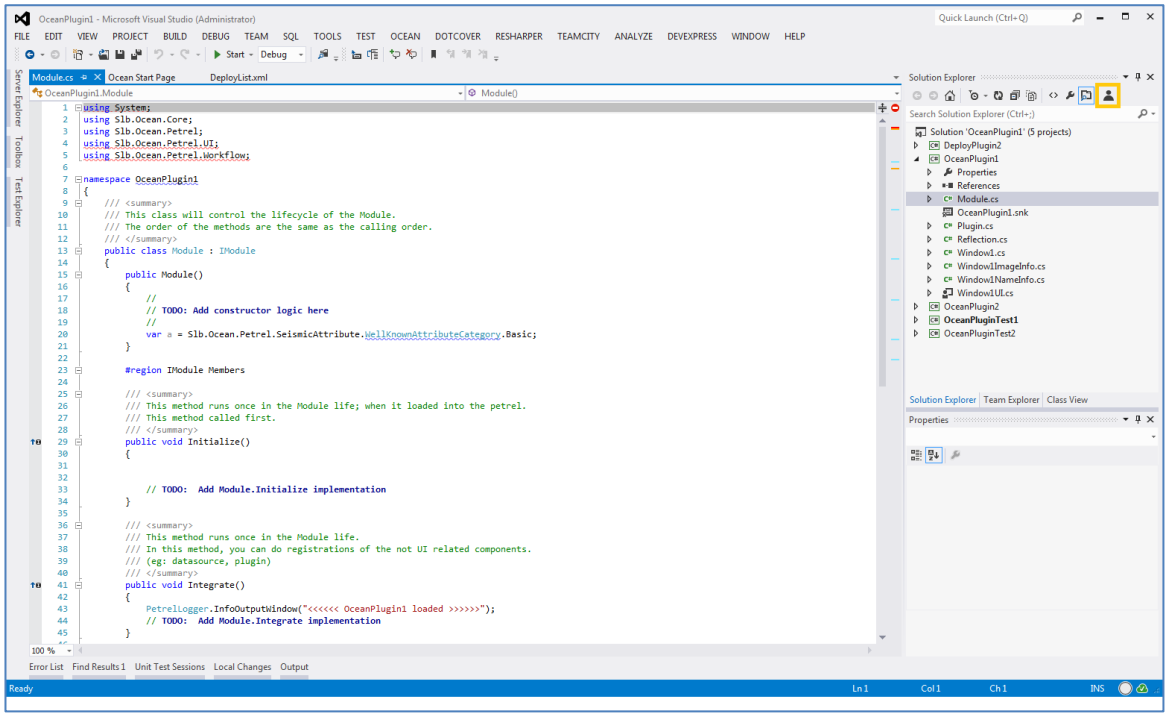


Figure 65: Access Ocean Quality Assistant from Solution Explorer window

Accessing Ocean Quality Assistant Summary

In Application Details tab the execution status of Automated Acceptance Tests, Manual Acceptance Tests, and Sanity Unit Tests are displayed.

Run all tests

From Application Details tab it is possible to run Automated Acceptance Tests, and fixed Sanity Unit Tests with a single click. The results displayed in the Summary are updated accordingly.

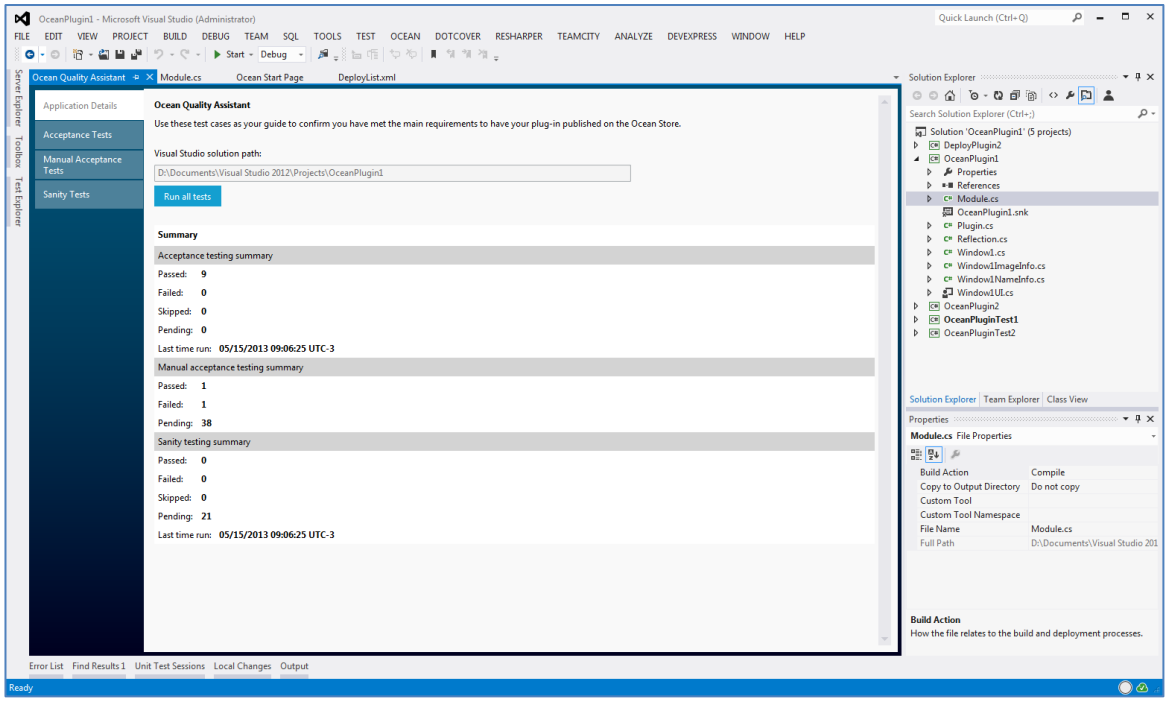


Figure 66: Application Details view

Running Automated Acceptance Tests

From Acceptance Tests tab it is possible to run a subset of tests that were originally performed only by the Ocean Acceptance Team.

Now plug-in developers can save time running this subset of tests before submitting the plug-in to Ocean Store.

Setup:

1. Select the *.pip or *.msi file of your plug-in;
2. Select the Ocean version against which you want to test your plug-in;
3. [Optional] Select a temporary folder where the *.pip or *.msi will be extracted during the tests. Otherwise the default temporary system folder will be used;
4. Click Run tests.

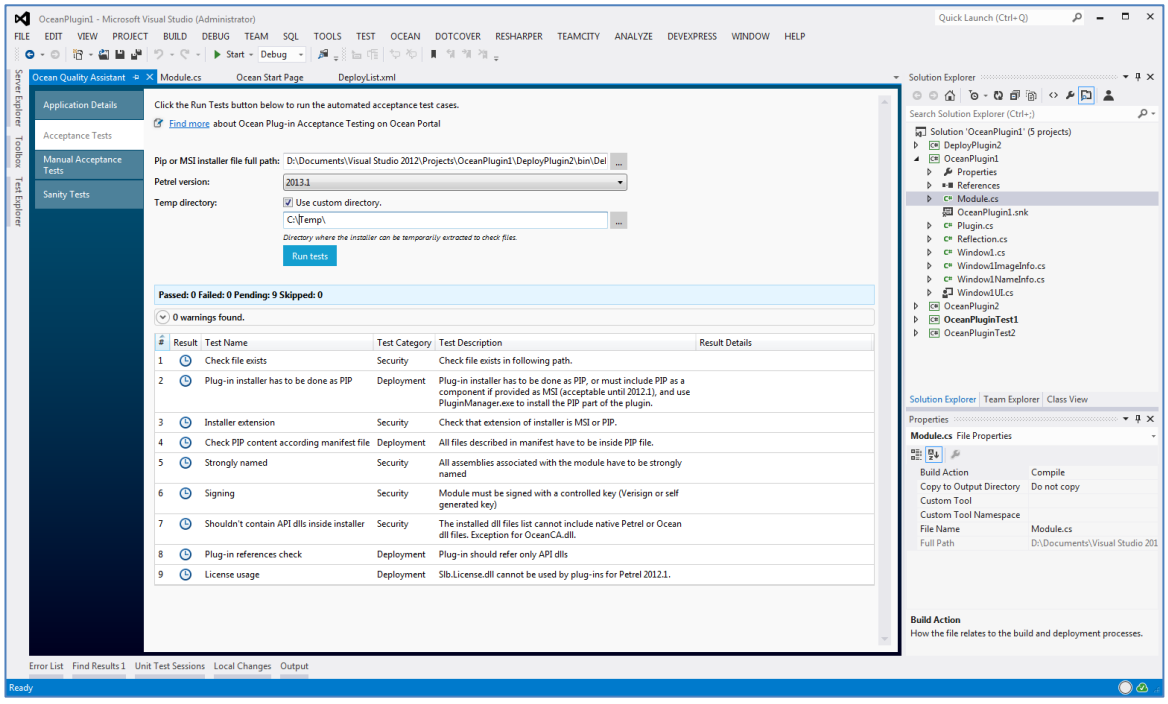


Figure 67 Acceptance Tests view

After the tests are run the result of each test is updated in the table, and a summary is shown below the Run tests button.

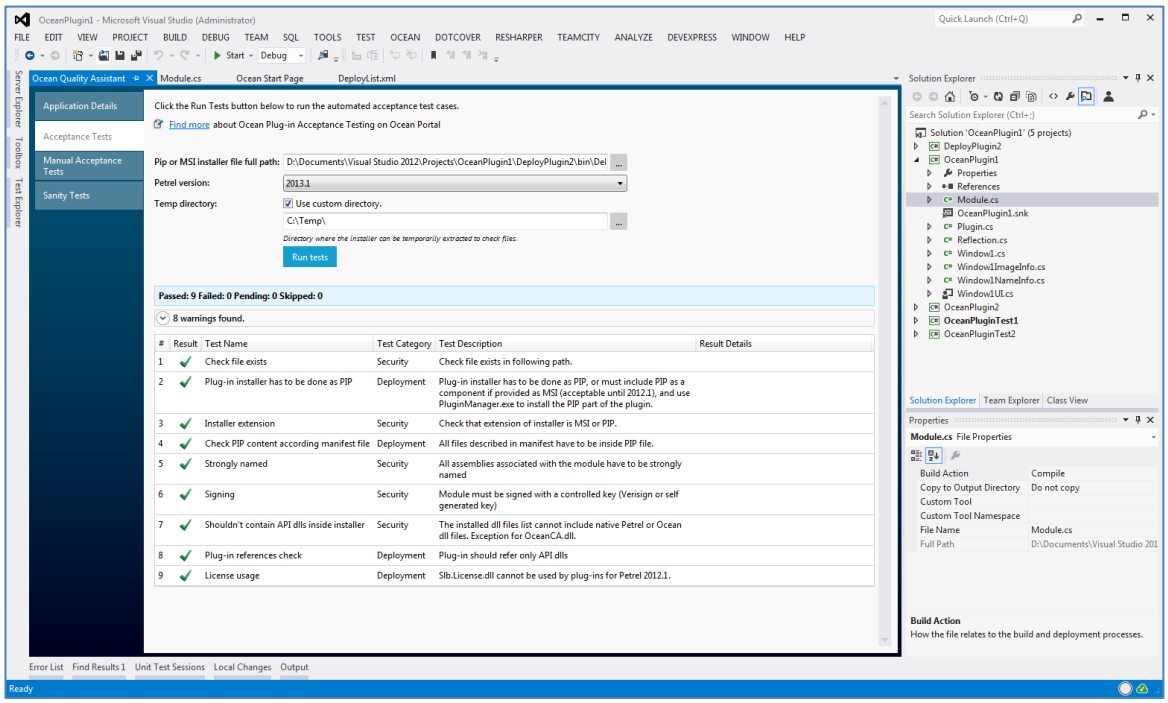


Figure 68 Acceptance Tests view after running complete

Internal Ocean APIs

During the execution of Acceptance Tests additional tests are executed to verify the occurrence of uses of internal Ocean APIs and calls by reflection.

Those occurrences are listed in the warning box between the Acceptance Tests table and the test execution summary.

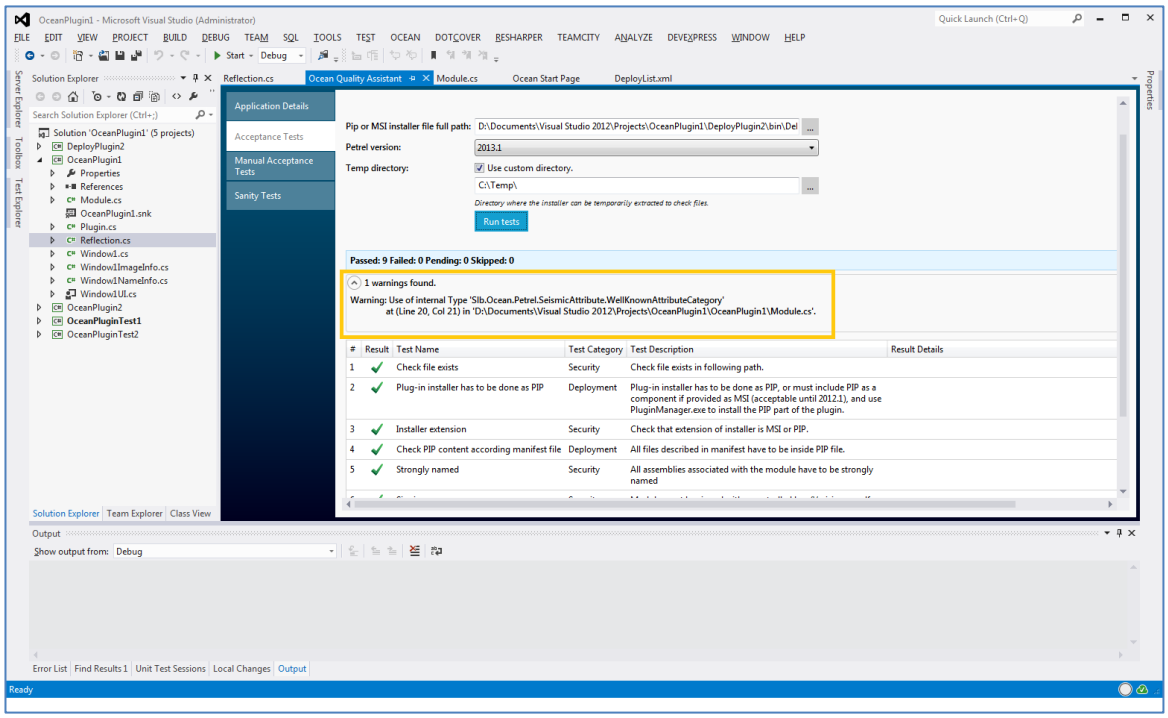


Figure 69 Warning box

Managing Manual Acceptance Tests

From Manual Acceptance Tests tab one can,

- Visit the list of acceptance criteria that have not been automated;
- Mark criteria as Pending, Passed, or Failed;
- Save and retrieve the results of the analysis of acceptance criteria;

Note: As the recorded result is persisted in text/xml file, it is easy to share and use version control on these results.

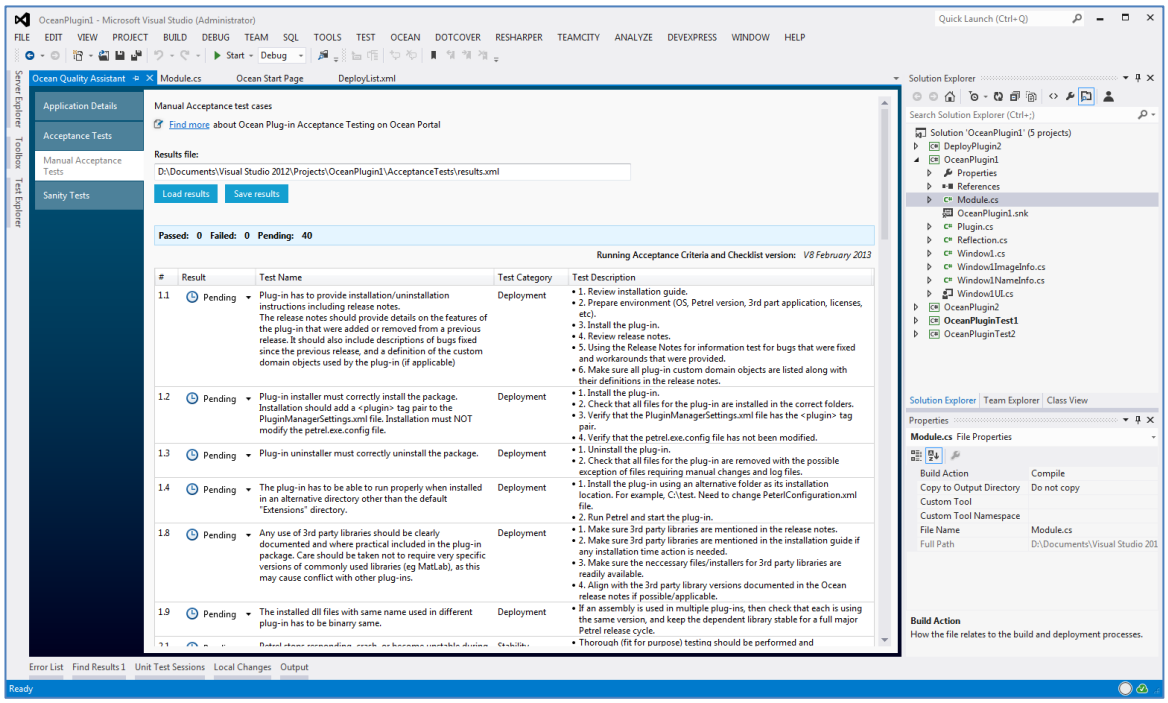


Figure 70 Manual Acceptance Tests view

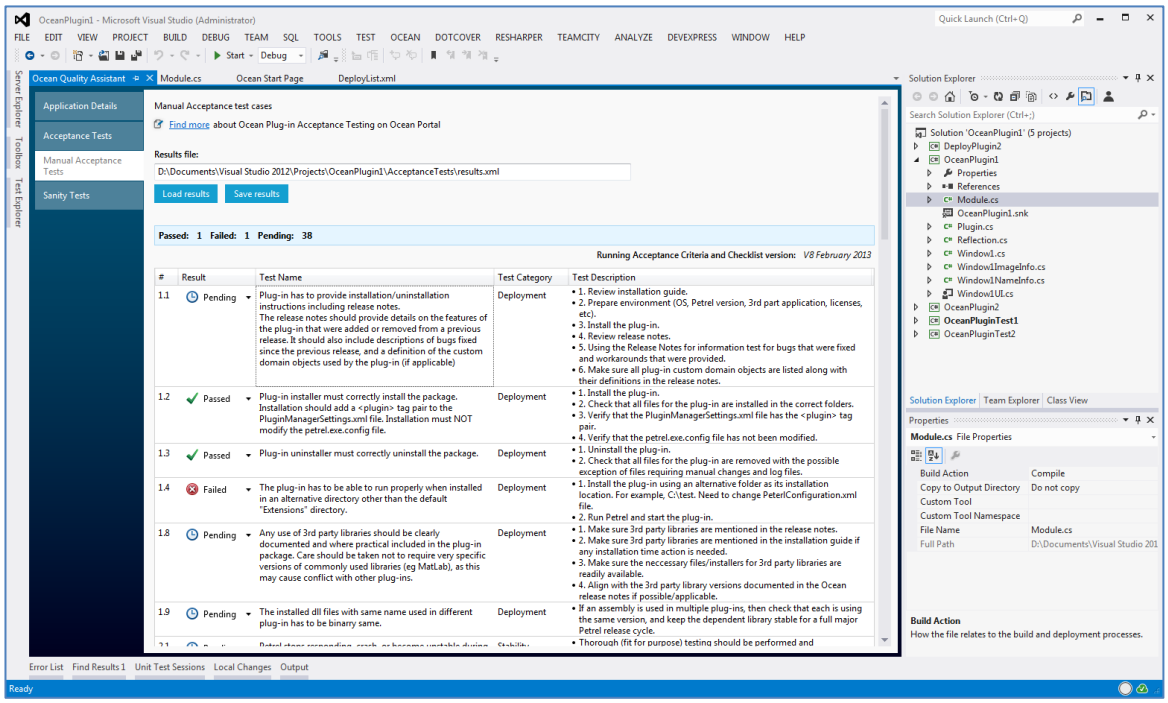


Figure 71 Manual Acceptance Tests view after saving edited results

Running fixed set of Sanity Unit Tests

From Sanity Tests one can run the fixed set of tests defined in Slb.Ocean.Petrel.Testing.SanityTest.dll (see Ocean for Petrel CHM > References > Walkthroughs > Sanity Unit Testing)

Setup:

1. Select a Petrel version;
2. Enter your plug-in namespace;
3. Select a Petrel project that includes the test target objects (custom domain objects, toggle windows, file formats, custom data sources etc.);
4. Click Run tests.

Note 1: Only installed Petrel and supported versions will be listed in Petrel version combobox.

Note 2: All test target objects have to be resolvable in the test project, and all corresponding plug-ins have to be registered.

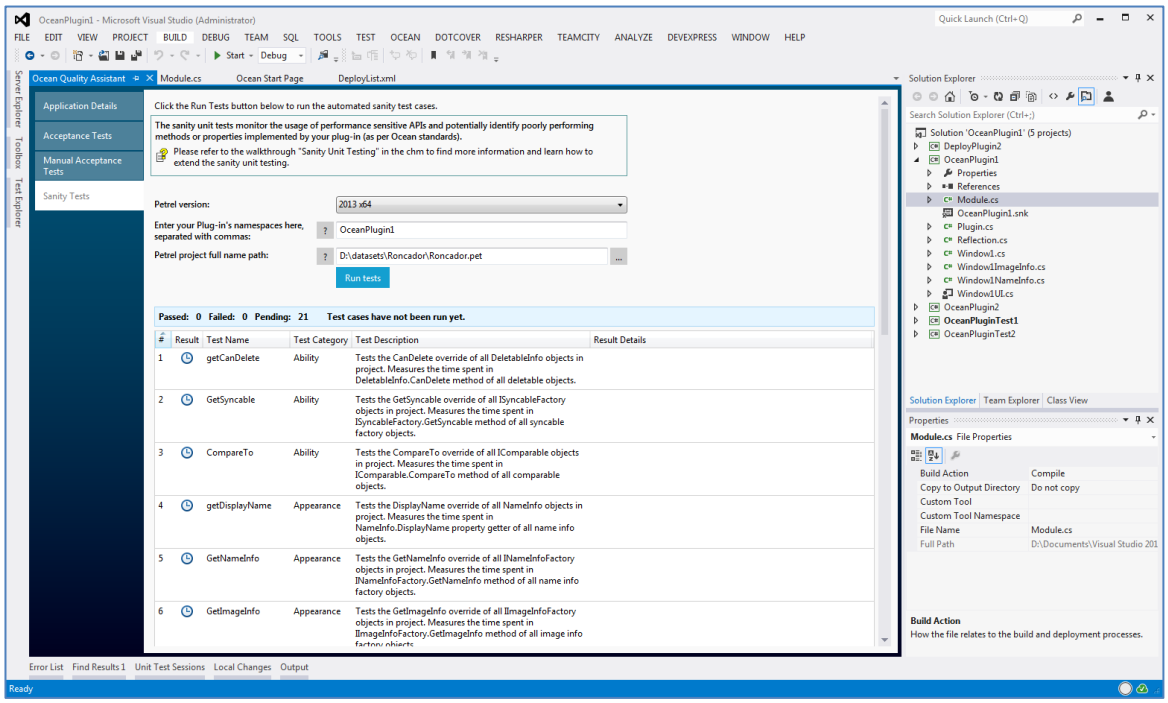


Figure 72 Sanity Tests view

During the execution of the tests, a status message is show below the Run tests button, and the log is displayed in the Output window.

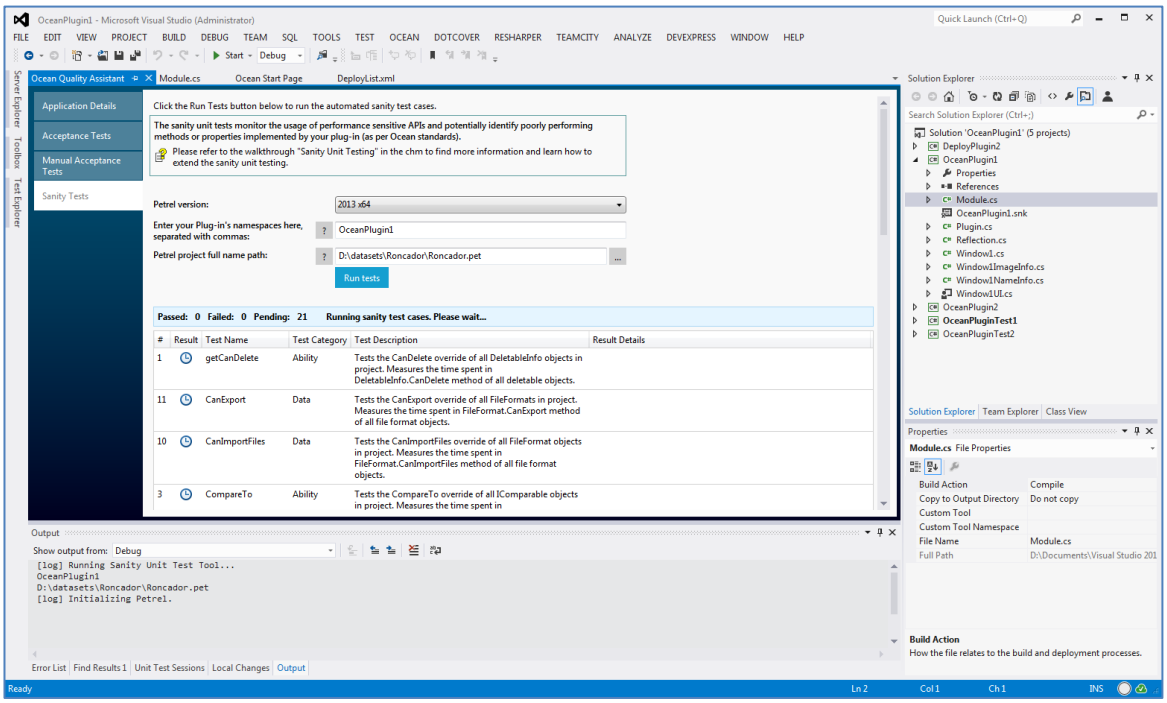


Figure 73: Sanity Tests view while executing test cases

After the tests are run the result of each test is updated in the table, and a summary is shown below the Run tests button.

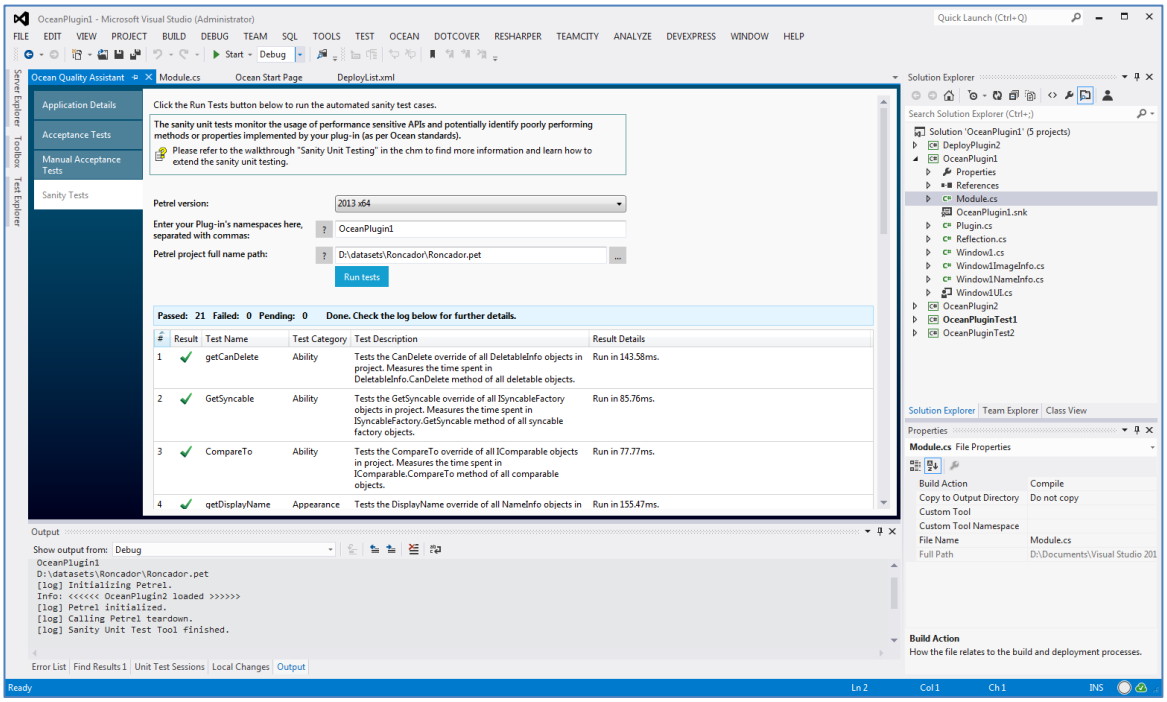


Figure 74 Sanity Tests view after finished execution

Ocean NUnit Test Adapter

Ocean NUnit Test Adapter for Visual Studio 2015 (VS2015) and Visual Studio 2017 (VS2017) is a test adapter that enables Ocean developers to seamlessly run integration tests using Ocean Testing Framework in Visual Studio.

This test adapter can be install separately after Ocean for Petrel SDK installation. Get the installer from \\Program Files\Common Files\Schlumberger\Ocean\Petrel\Common components\Visual Studio Extensions\Slb.OceanSdk.Petrel.TestAdapter.vsix.

In the following sections we will present the adapter's usage and the steps necessary to have these tests executed.

For further information about how to use Ocean NUnit Test Adapter in Team Foundation Server 2012 (TFS 2012) see "Testing Framework" in Ocean Help.

Creating a Test Project

First you will need a plug-in and a test project that will be used as example to show the features of the Ocean NUnit Test Adapter. These projects can be generated using the Ocean Wizards in Visual Studio 2015/2017.

In the following steps, we will use two projects as example: OceanPlugin and OceanPlugin.Tests, an Ocean plug-in and a test project respectively.

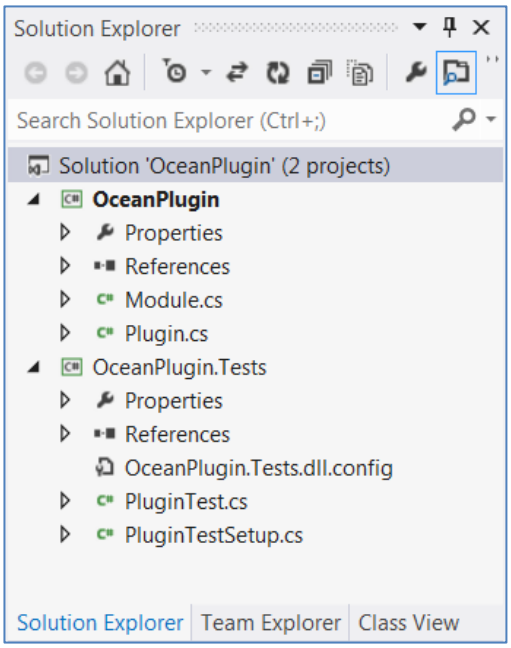


Figure 75 Solution with an Ocean plug-in and its test project.

Test Explorer

With your test project created, the next step is to open the Test Explorer window in VS2015/VS2017. This can be accomplished by selecting the menu TEST > Windows > Test Explorer. As result, a window will be displayed and it will be responsible to manage your tests.

After you build the projects, the tests should be displayed in the Test Explorer as depicted in Figure 76.

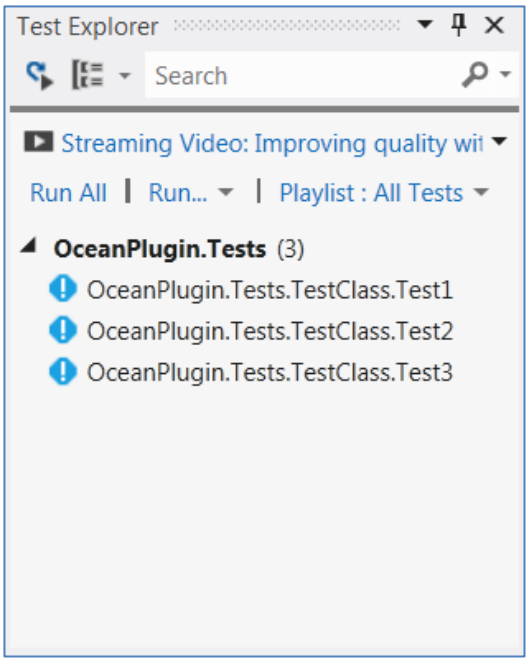


Figure 76 Tests displayed in the Test Explorer window

Note that this is a native VS2015 window and there are several run options and filters like Run failed tests, Run Not Run tests, and others.

Selecting Petrel Version

Before running the tests, it is important to select the correct Petrel version and platform (Figure 77), because the test adapter needs to be aware of which Petrel will be used in order to infer the correct Petrel installation path.

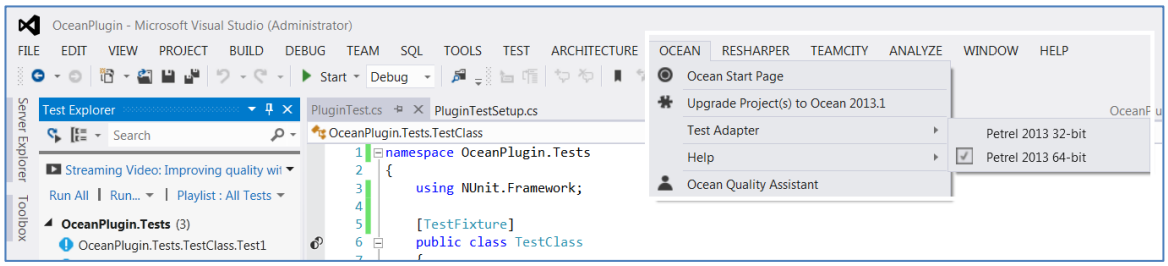


Figure 77 Selecting the Petrel version and platform

Ocean NUnit Test Adapter requires the test assembly to be located inside the Petrel's root folder in order to resolve the dependencies. For this reason, besides selecting the correct Petrel version and platform, it is necessary to set the projects' outputs to Petrel's root folder.

Running the tests

After configuring the test project and the test adapter, the user should be able to run the tests by using the Test Explorer window commands (i.e. Run All and Run…) and this will trigger the test execution. The test adapter's output can be visualized in the Output window when the option "Tests" is selected in the "Show output from:" combo box.

Figure 78 depicts an example of test output. The first section of the output just verifies if the requirements to use the Ocean Testing framework are met. This will include the verification of the existence of the PetrelConfiguration.xml file, list the installed plug-ins, and others.

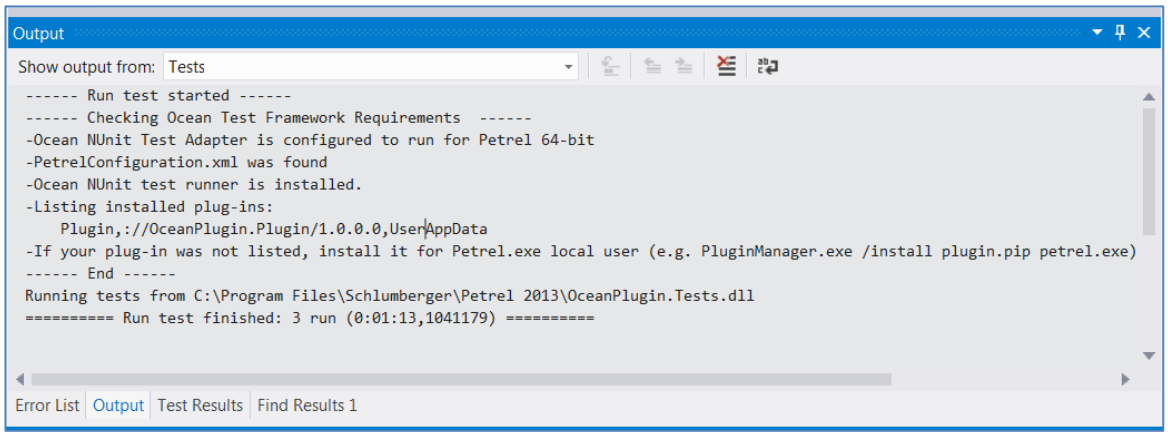


Figure 78 Tests output

It is important to watch this output, because if anything goes wrong, it will give some advice on how to execute the tests properly or explain why the Ocean Testing Framework could not be ran.

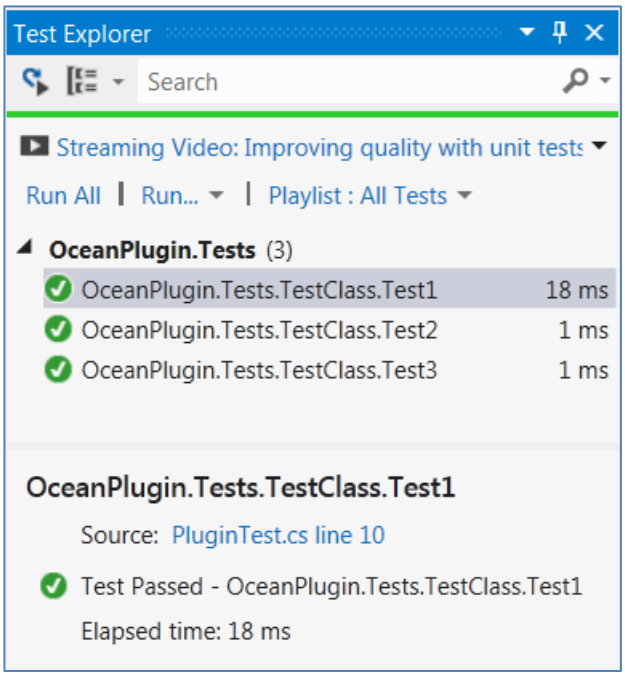


Figure 79 Test Explorer with the tests results

After the test execution finishes, the tests are updated in the Test Explorer with the details of each test case (Figure 79).

Code coverage

Test Explorer also supports code coverage analysis. This can be accessed by running the commands “Analyze Code Coverage for All Tests” or “Analyze Code Coverage for Selected Tests” in the window. The result will be displayed as depicted in Figure 80.

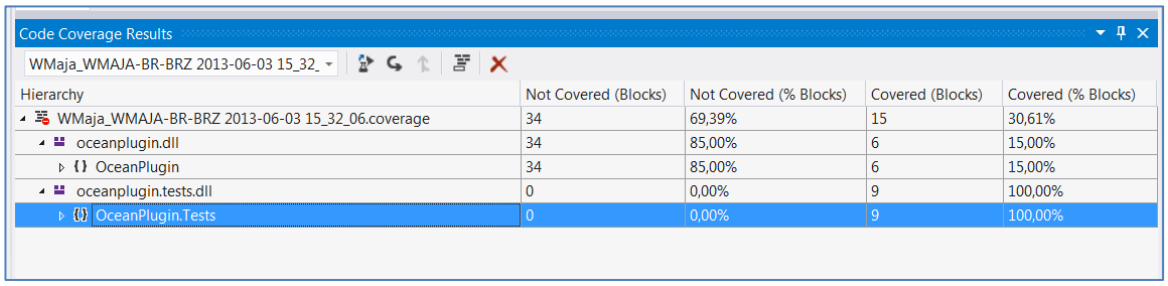


Figure 80 Code Coverage result

The source code will also be colored with respect to the coverage results.

Ocean Plug-in Upgrade Tool

Ocean Plug-in Upgrade Tool is a Visual Studio Extension that upgrades assembly references to the current version given a C# Plug-in project.

Running the Upgrade Tool

In Visual Studio, Ocean Plug-in Upgrade Tool can be accessed through, either

- VS main menu > Ocean > Upgrade Project(s) to Ocean 201X.1, or
- Solution Explorer > Right click one **Solution** > Upgrade Project(s) to Ocean 201X.1, or
- Solution Explorer > Right click one **Project** > Upgrade Project(s) to Ocean 201X.1, or
- Solution Explorer > Right click **multiple Projects** > Upgrade Project(s) to Ocean 201X.1

When the Ocean Upgrade Tool is accessed through Ocean Menu or Solution context menu **all loaded projects under the opened solution** will be upgraded.

Otherwise, only the selected project(s) will be affected.

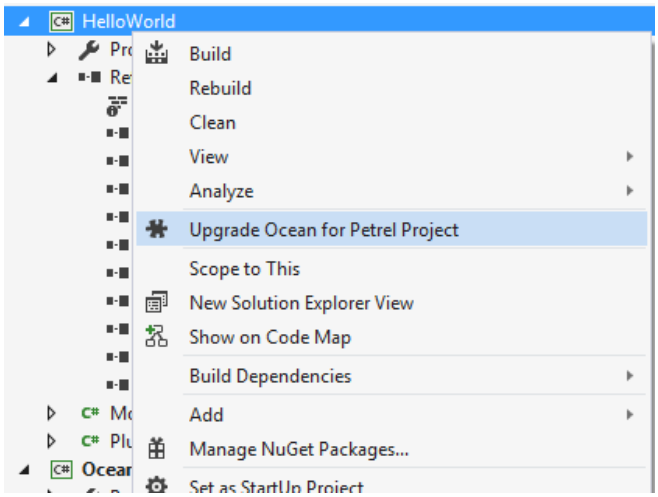


Figure 81: Accessing Ocean Plug-in Upgrade Tool through Visual Studio Project context menu

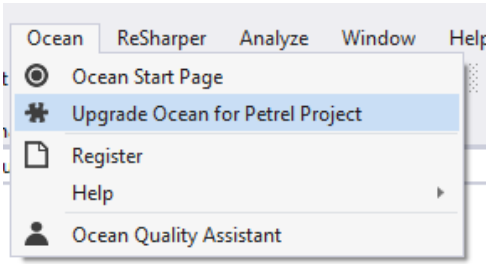


Figure 82 Ocean Plug-in Upgrade Tool is also available in the OCEAN Menu

What to expect

- After selecting the “Upgrade Project(s) to Ocean 201x.1” a confirmation dialog will be shown;
- After selecting “Yes” in the confirmation dialog, the upgrade process will be started over the selected project(s) or solution;

- A processing dialog will be displayed while the necessary changes are being made;
- The *.CSPROJ and *.CSPROJ.USER files will be backed up under the same directory of theirs originals;
- Log messages will be shown in Visual Studio Output window.

Remarks:

1. During the last phase of the upgrade the projects will be reloaded;
2. The Upgrade Tool will affect only Ocean and Open Inventor assembly references;
3. No source code is modified;
4. The user might remove obsolete API calls manually.

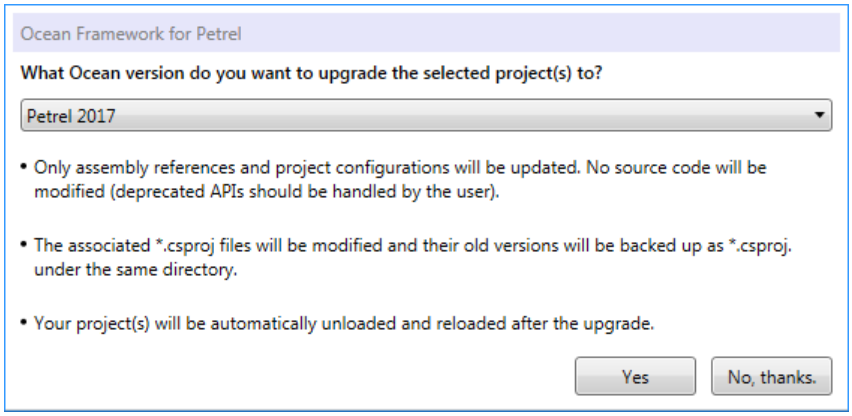


Figure 83 Confirmation dialog

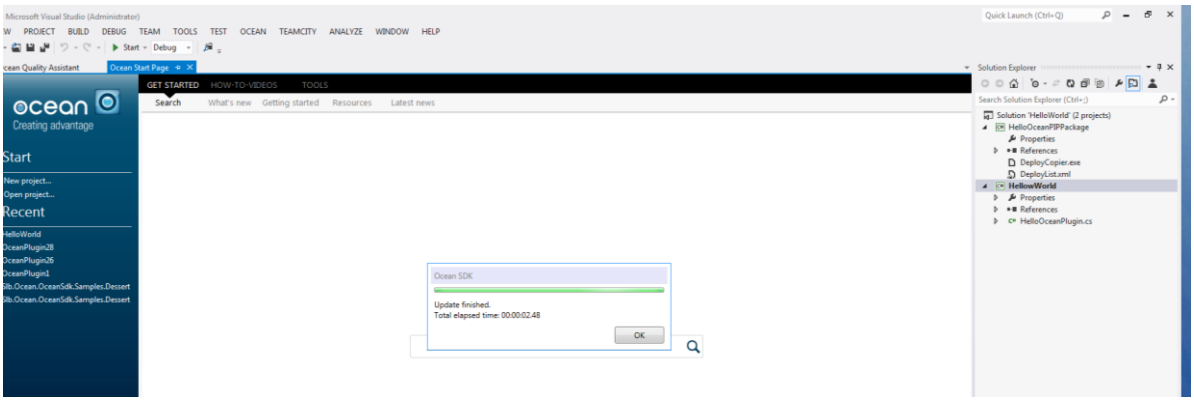


Figure 84 Processing dialog and log messages

Ocean for Petrel API Documentation in Help Viewer

Ocean API documentation is accessible in Microsoft Help Viewer. Ocean API documentation content in Help Viewer is consistent with Ocean for Petrel CHM. To enable API documentation access code, set help preference to launch in help viewer as below:

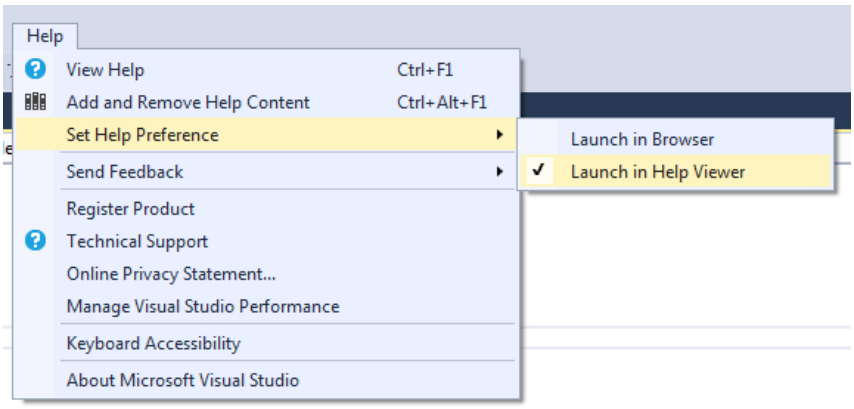


Figure 85 Setting help preference

This will enable access to any API documentation from code editor by placing cursor on the API member and pressing F1 key.

Ocean Controls

Ocean controls are available in Slb.Ocean.Petrel.UI.Controls.dll from Petrel installation folder. Add them to your Toolbox using Choose Toolbox Items dialog.

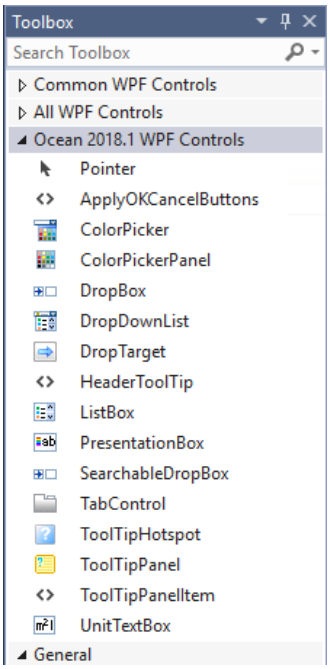


Figure 86 Ocean Controls