# Retrieve PI Points and Edit Point Attributes using the PI SDK

OSIsoft.

## 1.1 Retrieve PI Points and Edit Point Attributes using the PI SDK

### 1.1.1 Description

Learn to use the PI SDK to retrieve PI Points and Edit Point Attributes.

### 1.1.2 Objectives

- Retrieve points (tags) that correspond to a point mask;
- Retrieve points using the **Tag Search** window;
- Use the **PointList** collection.
- Display the properties of a PI Point via a custom list;
- Display the properties of a PI Point via the **Point Attributes** window;
- Modify the value of a point's properties;
- Add a PI Point.

### 1.1.3 Problem Description

You need to develop a custom application and give your end-users the possibility to retrieve points based on a tag mask or use the **Tag Search** dialog window. Here are the specifications required for your application:

- When the **Search** button is clicked, build the `WHERE` clause with the specified tag mask and search for corresponding points on the server;
- Loop through all the **PIPoint** objects of the list, and populate a list box on the form;
- Add a **Picker** button (… button) that will call the **Tag Search** dialog box. You will need to configure the **Picker** to allow a single PI Server in the selection.
- Get the list of all the attributes of a selected PI Point;
- Update the value of a point attribute;
- Add a button that will display the standard **Point Attributes** window;
- Add a functionality to let the user create a new PI Point;
- Add a check box field that will create a random tag using these attributes: **pointsource**=`R`, **location2**=`50`, **location4**=`1` and **location5**=`1`. A random PI Point is point configured in such a way the PI Random Simulator will generate simulation values for it.

### 1.1.4 Approach

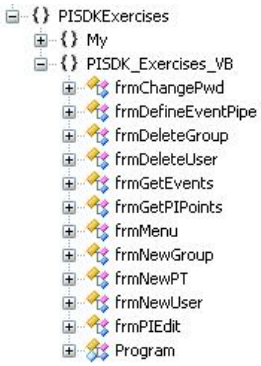
Using the Visual Studio template project found in `C:\Labs\Data Access\Retrieve PI Point and Edit Point Attribute\Template\PISDK_Exercises.sln`

complete the application foundation with the methods and properties required to perform the requested tasks. This solution contains a copy of the project in VB and C#. To facilitate your work, all the accessory development (UI, naming convention, initialization, etc.) has been

done for you. Locate placeholders in the code to know where to put your code; they look like this: `'<Complete the code here>.`

A class view of the `PI SDK_Exercises_VB` namespace can be seen to the figure beside. The main entry point of this application is made to the `Main` method of the `Program` class.

For this exercise, you will have to modify the `frmGetPIPoints, frmPIEdit` and the `frmNewPT` classes.




For all parts of this exercise, you will need to refer to the **OSIsoft.PISDK, OSIsoft.PISDKCommon** and **OSIsoft.PITimeServer** namespaces.

**Part A**

- Using the `frmGetPIPoints` form, when the `uxSearchButton` button is clicked, form a `WHERE` clause using the user-specified tag mask.
- Using the `WHERE` clause, retrieve corresponding PI Points from `_PIServerRef`.

**Part B**

- For each PI Point in the list of retrieved PI Points, add an entry in the combo box `uxAvailablePIPointsList` on the `_frmPIEdit` form.

**Part C**

- On the `frmPIEdit` form, when the `uxTagSearchButton` button is clicked, use the `_DlgTagSearch` object to display the **Tag Search** dialog window.
- Have the **Tag Search** dialog set to initialize the search with the current server and set so that the user cannot select other servers.
- Add the PI Points retrieved to the `uxAvailablePIPointsList` combo box.

**Part D**

- Assign the PI Point selected in the `uxAvailablePIPointsList` combo box to the `_CurrentPIPoint` object when the selected index is changed.

**Part E**

- When the `uxGetAttributesButton` button is clicked, retrieve the list of attributes for the selected PI Point.
- For each attribute retrieved, add an entry in the `uxPointAttributesList` list box.

**Part F**

- When an attribute in the list box is selected, display its value in the `uxAttributeValue` text box.

**Part G**

- When the `uxChangeAttrValueButton` button is clicked, update the value of the selected attribute to the value the used specified.

**Part H**

- When the `uxAttributesWindowButton` button is clicked, use a **PISDKDlg.PIPointPropertySheet** object to display the **Point Attributes** window for the selected PI Point.

**Part I**

- When the `uxAddPointButton` button is clicked on the `frmNewPT` form, create a new PI Point.
- The new PI Point will have the user specified name and point type.
- If the check box for a random tag is **not** checked, create the PI Point with the **Classic** point class.
- If the check box for a random tag is checked, create the PI Point with the **Classic** point class with the following attributes: **pointsource** = R, **location2** = 50, **location4** = 1 and **location5** = 1.
- Update the list of PI Points on the main `frmPIEdit` form.


Try to do this exercise on your own before proceeding to the step-by-step solution.

### 1.1.5 Step-by-Step Solution

**Part A**

1. In the `frmGetPIPoints` class, find the `uxSearchButton_Click` event.
2. Build a string with the entered tag mask.

```
WhereClause = "Tag = """ & uxTagMask.Text & """"
```

**Note:** When you need to enter a double quote character within a string already delimited by double quote with Visual Basic .NET, you need to enclose it with two (2) double quote characters like this `"""`. If you need to store only one double quote character into a string object you need to use four (4) double quote characters in a row like this: `""""`.

3. Define a **PointList** object.

```
Dim List As PointList
```

4. Use the **GetPoints** method to retrieve PI Points with the `WHERE` string

```
List = _frmPIEdit._PIServerRef.GetPoints(WhereClause)
```

5. Remove the condition True = True and replace it with the **PointList.Count** property.

```
If List.Count = 0 Then
```

**Part B**

1. Loop through all the retrieved PI Points and add entries to the combo box `uxAvailablePIPointsList` on the main `frmPIEdit` form.

```
For Each Point as PIPoint in List
    _frmPIEdit.uxAvailablePIPointsList.Items.Add(Point.Name)
Next
```

**Part C**

1. In the `frmPIEdit` class, find the `uxTagSearchButton_Click` event.
2. Add the current PI Server to a collection.

```
PIServersCollection.Add(_PIServerRef.Name, "")
```

3. Display the **Tag Search** window using the `_DlgTagSearch` object and return the selected points
   - Use the `tsoptDisableServerPickList` option to prevent the user from searching another PI Server

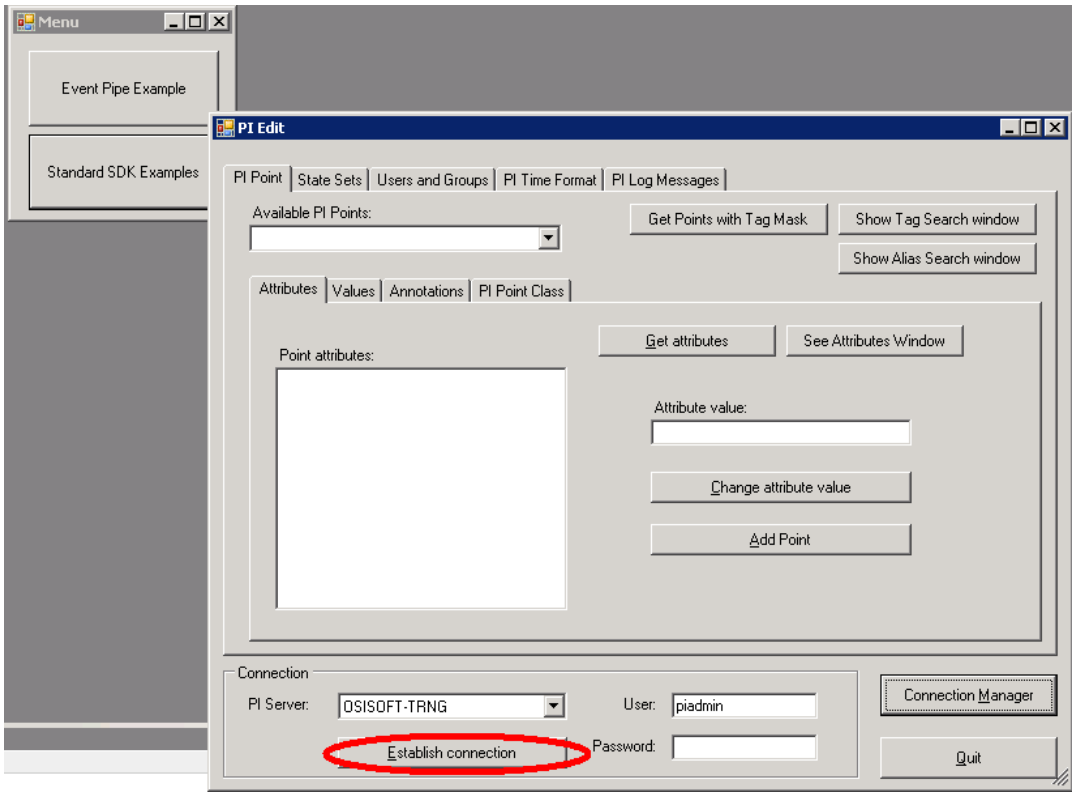
```
Points = _DlgTagSearch.Show(PIServersCollection, _
                PISDKDlg.TagSearchOptions.tsoptDisableServerPickList)
```

4. Loop through all of the retrieved PI Points and add an entry for each to the combo box.

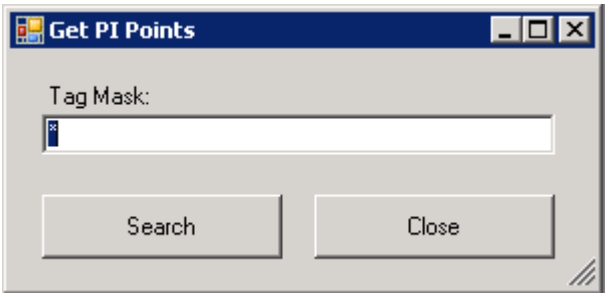
```
For Each Point As PISDK.PIPoint In Points
    'Add the name of the PI Point to the combo box.
    uxAvailablePIPointsList.Items.Add(Point.Name)
Next
```

**You can now test the application by retrieving a PI Point:**

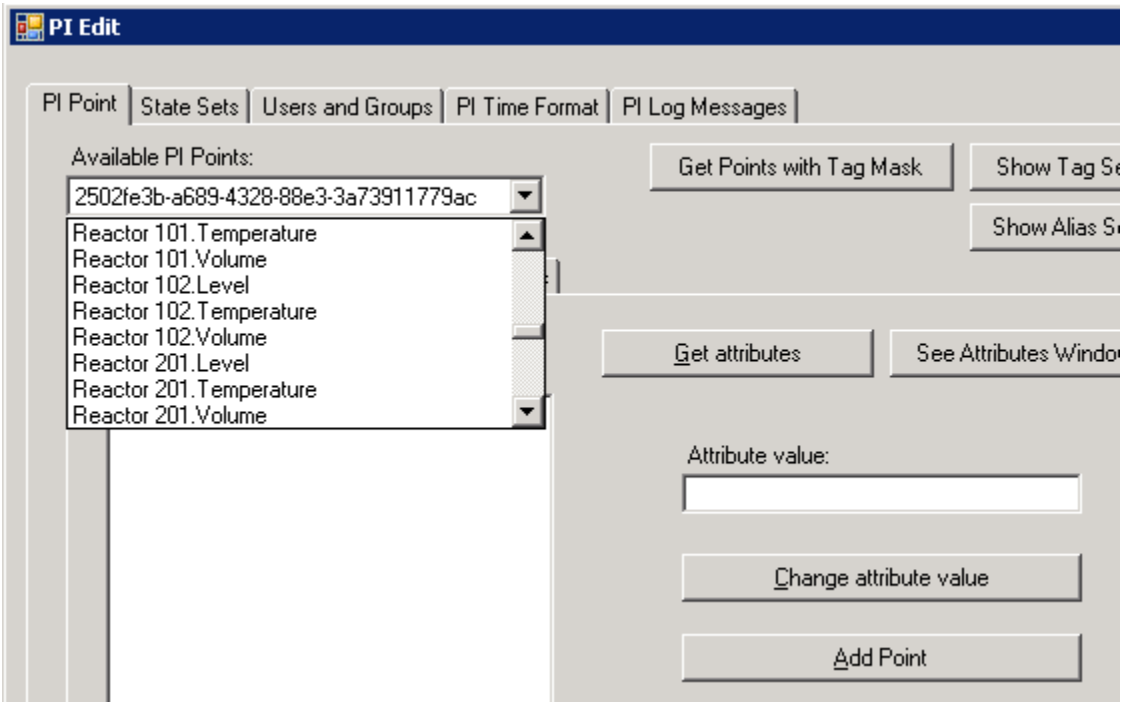
1. Run the application
2. Select “Standard SDK examples”
3. Click on “Establish connection”




4. Click on “Get Point with Tag Mask”




5. Click on Search
6. Examine the result under “Available PI Points”




**Part D**

1. Use the point name selected in the combo box to retrieve the desired PI Point from the current server. Assign this point to the `_CurrentPIPoint` object in the `uxAvailablePIPointsList_SelectedIndexChanged` event.

```
_CurrentPIPoint = _PIServerRef.PIPoints(uxAvailablePIPointsList.Text)
```

**Part E**

1. Add an entry in the point attribute list box for each point property in the attribute list for the selected PI Point in the `uxGetAttributesButton_Click` event.

```
For Each Attribute As PISDK.PointAttribute In _CurrentPIPoint.PointAttributes
    uxPointAttributesList.Items.Add(Attribute.Name)
Next
```

**Part F**

1. Get the value of the selected attribute and update the value of the text box in the `uxPointAttributesList_SelectedIndexChanged` event.

```
uxAttributeValue.Text = _
    _CurrentPIPoint.PointAttributes.Item(uxPointAttributesList.Text).Value.ToString
```

**Part G**

**Note:** It exists a better method to modify attributes of a PI Point with a single call to the PI Server. This will be covered later.

1. Find the `uxChangeAttrValueButton_Click` event.
2. Allow property modification by disabling the **ReadOnly** property on the PI Point attributes in the `uxChangeAttrValueButton_Click` event.

```
_CurrentPIPoint.PointAttributes.ReadOnly = False
```

3. Modify the selected property by assigning it the specified value.

```
MyAttributeList.Add(uxPointAttributesList.Text, uxAttributeValue.Text)
_CurrentPIPoint.PointAttributes.ModifyAttributes(MyAttributeList, MyErrorList)
```

4. Remove the property modifications rights.

```
_CurrentPIPoint.PointAttributes.ReadOnly = True
```

**Part H**

1. Find the `uxAttributesWindowButton_Click` event.
2. Add the current PI Point to a **PointList** collection.

```
Dim SelectedPoint As New PISDK.PointList

SelectedPoint.Add(_CurrentPIPoint)
```

3. Display the **Point Attribute** dialog window as a modal window.

```
DlgAttributes.Show(SelectedPoint, 1)
```

**Part I**

1. In the `frmNewPT` class, find the `uxAddButton_Click` event.
2. Retrieve the specified point name.

```
PointName = uxPointName.Text
```

3. Detect the user specified point type.

```
If uxPointTypeOption1.Checked = True Then PointType =
PointTypeConstants.pttypFloat16
If uxPointTypeOption2.Checked = True Then PointType =
PointTypeConstants.pttypFloat32
If uxPointTypeOption3.Checked = True Then PointType =
PointTypeConstants.pttypFloat64
If uxPointTypeOption4.Checked = True Then PointType =
PointTypeConstants.pttypInt16
If uxPointTypeOption5.Checked = True Then PointType =
PointTypeConstants.pttypInt32
If uxPointTypeOption6.Checked = True Then PointType =
PointTypeConstants.pttypString
```

4. If the random tag box is **not** checked, create the point using the specified name and type with the **Classic** point type.

```
If uxRandomTagCheck.Checked = False Then
```

```
    PointReturned = _frmPIEdit._PIServerRef.PIPoints.Add(PointName, "Classic", _
                    PointType)
...
```

5. If the random tag box is checked, populate a collection with the specified attributes.

```
Else
    Attributes.Add("pointsource", "R")
    Attributes.Add("location2", 50)
    Attributes.Add("location4", 1)
    Attributes.Add("location5", 1)
```

6. Create the point using the previously specified name, type, point class **Classic**, and the created attributes.

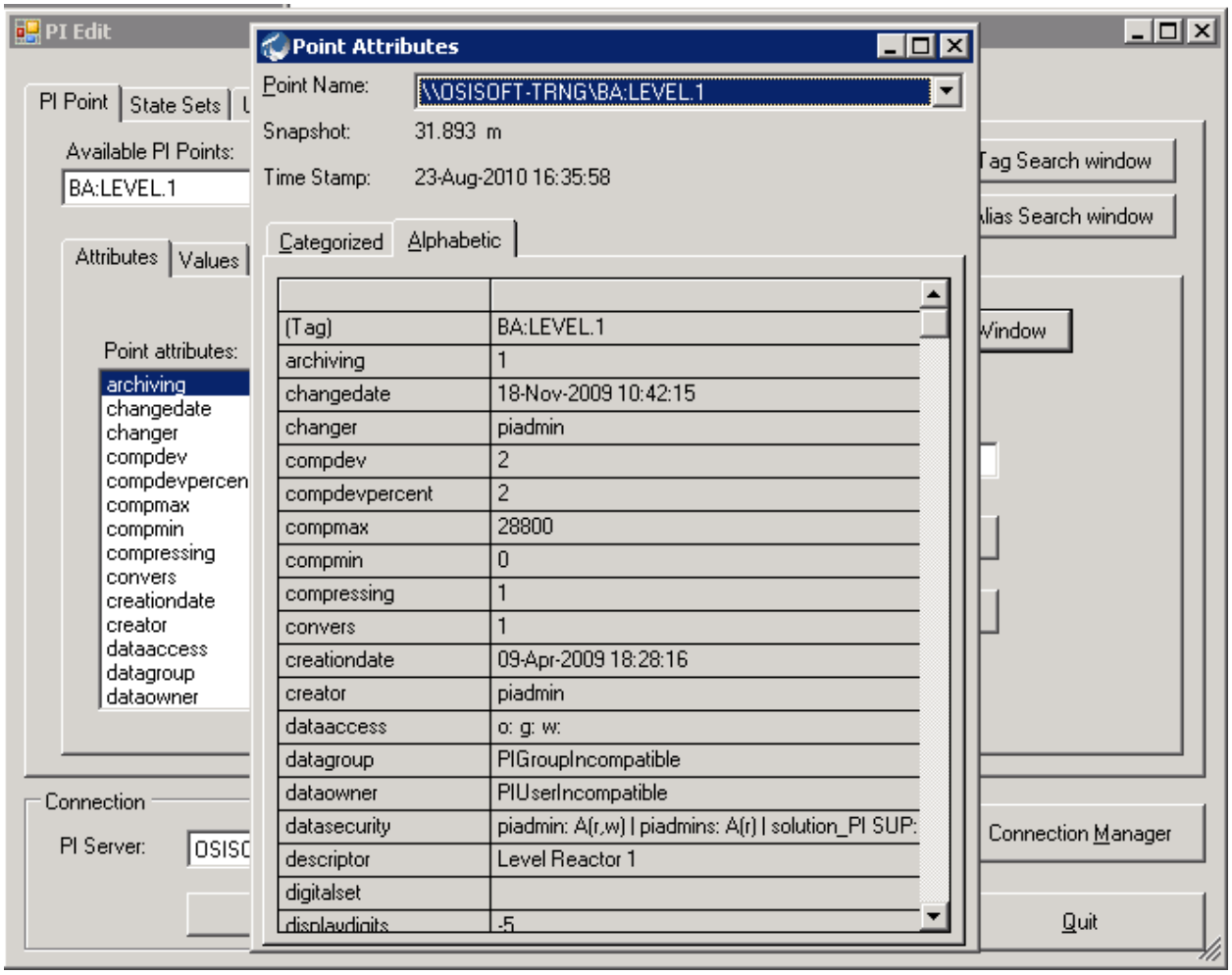
```
    PointReturned = _frmPIEdit._PIServerRef.PIPoints.Add(PointName, "Classic", _
                    PointType, Attributes)

End If
```

7. Update the PI Points combo box on the main `frmPIEdit` form from the `frmNewPT` one.

```
_frmPIEdit.uxAvailablePIPointsList.Items.Add(PointName)
_frmPIEdit.uxAvailablePIPointsList.Text = PointName
```

**You can now test the application by retrieving a PI Point:**

1. Run the application
2. Select “Standard SDK examples”
3. Click on “Establish connection”
4. Click on “Get Point with Tag Mask”
5. Click on Search
6. Select a PI Point from the list
7. Click on Get Attributes, and the attributes will be displayed on the Point Attributes window
8. Click on See Attributes Window and a new window will open




9. Close the window
10. Change the attribute for archiving being displayed from 1 to 0
11. Click on Add Point
12. Type a name for the new Point and click Add
13. Search for the available points again and verified that the new Point was created