Using Event Pipes in PI SDK
1.1 Using Event Pipes in PI SDK

1.1.1 Objectives

At the end of this learning lab, you should be able to:

- Create an EventPipe object that queues changes to the value of a PI tag.
- Take values from that queue one at a time,
- Take all values from the queue at once.
- Configure an event handler that triggers when the tag receives a new snapshot value.

In this exercise a connection to the server "srv" and access to the PI point "CDT158" through the variable _PIPoint is already implemented in the template.

1.1.2 Problem Description

Create an application that creates an EventPipe object for the tag "CDT158" that queues changes to the value of that tag. Add a button to "take" values from that queue one at a time. Add another button that takes all values from the queue at once. Finally, configure an event handler that allows you to trigger the display the new value of the tag when it receives a new snapshot value.

1.1.3 Approach

To do this exercise, you will need to refer to the OSIsoft.PISDK, OSIsoft.PISDKCommon and OSIsoft.PITimeServer namespaces.
Reference PI SDK Libraries

Different assemblies are required to include with your project to develop with the PI Server; you do not necessarily need all of them at all time but it is a good practice to have them ready to use. These assemblies are all installed in the GAC, so you can reference them in your project by simply browsing the list and pick the ones described in the following list:

- OSIsoft.PISDK.dll is the main PI SDK dynamic link library (DLL) – primary library for PI SDK content;
- OSIsoft.PISDKCommon.dll is used for some shared data structures;
- OSIsoft.PITimeServer.dll provides parsing capabilities of time strings and correct rendering of time in several useful forms and time arithmetic;
- OSIsoft.PISDKDlg.dll contains standard PI SDK functions like tag search and connection handling;
- OSIsoft.PISDKCtl.dll contains standard PI SDK controls.

Part A -
Get an EventPipe for tag CDT158 and take values one at a time

Part B -
Use TakeAll to retrieve all values in the EventPipe

Part C -
Create an event handler to utilize EventPipe events to trigger retrievals

Part D -
In the newly created OnNewValue event of the event handler take values from the queue when a new event is added to the queue.

Part E -
Use the Archive Edit utility in PI-SMT to write values to CDT158 and thereby test the applications ability to display updates to the value of the tag.

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

Part A - Get an EventPipe for tag CDT158 and Take Values One at a Time

In this exercise a connection to the server "srv" and access to the PI point "CDT158" through the variable _PIPoint is already implemented in the template.

1. In the btnSubscribe_Click method: get access to the EventPipe property of the Data property of the PI point "CDT158" through the variable defined as PISDK.PIPoint.

   \[
   \text{cdtEvents} = \text{PISDK.PIPoint.Data.EventPipe}
   \]

2. In the btnTake_Click method: Use the `Take` method of the variable that represents the EventPipe object. The results of the Take method are stored in a variable of type PISDK.PIEventObject.

   \[
   \text{_PIEventObject} = \text{cdtEvents.Take}
   \]

3. Assign the EventData part of the Event to a temporary PointValue object.

   \[
   \text{_pv} = \text{_PIEventObject.EventData}
   \]

Note: it is through the temporary variable _pv or PointValue type that you access the name, timestamp, and value of the returned events. This loop is already implemented in the template.

Example:

\[
\text{_lvi = New ListViewItem(_pv.PIPoint.Name.ToString())}
\text{_lvi.SubItems.Add(_pv.PIValue.TimeStamp.LocalDate.ToString())}
\text{_lvi.SubItems.Add(_pv.PIValue.Value.ToString())}
\]

Part B - Use TakeAll to Retrieve all Values in the EventPipe
1. In the btnTakeAll_Click method: Use the **TakeAll** method of the variable that represents the EventPipe object.

   ```csharp
   ReturnedEvents = cdtEvents.TakeAll
   ```

2. Set the variable defined as **PIEventObject** to the first returned event

   ```csharp
   _PIEventObject = ReturnedEvents(i)
   ```

3. Just as with the **Take** method, assign the EventData part of the Event to a temporary **PointValue** object. Likewise, use the same loop to retrieve the name, timestamp, and value of each event.

   ```csharp
   _pv = _PIEventObject.EventData
   ```

---

**Part C - Create an Event Handler to Utilize EventPipe Events to Trigger Retrievals**

1. Return to the btnSubscribe_Click method: Because the project is being written in a .NET environment, the COM based PISDK.EventPipe interface needs to be exposed through a specialized interface made available in a PISDK interoperability assembly. The event interface (interop) that corresponds to the EventPipe interface is **PISDK._DEventPipeEvents_Event**. At this point you must set the variable defined as type **PISDK._DEventPipeEvents_Event** to point to the eventpipe variable cdtEvents that you defined in Part A above.

   ```csharp
   epEventsCDT = cdtEvents
   ```

2. In the btnUpdate_Click you now must create the event handler. Create the event handler for InterfaceName_EventNameEventHandler.

   ```csharp
   Dim newValueHandler = New PISDK._DEventPipeEvents_OnNewValueEventHandler(AddressOf epEventsCDT_OnNewValue)
   AddHandler epEventsCDT.OnNewValue, newValueHandler
   ```

3. The btnUpdate_Click functions as a toggle turning on and off updating the display. So you must also remove the event handler.

   ```csharp
   Dim newValueHandler = New PISDK._DEventPipeEvents_OnNewValueEventHandler(AddressOf epEventsCDT_OnNewValue)
   RemoveHandler epEventsCDT.OnNewValue, newValueHandler
   ```

---

**Part D - Take Values from the Queue when a New Event is Added to the Queue.**

4. In the **epEventsCDT_OnNewValue** method: Just as you used the **Take** method in Part A above, use the Take method and the loop to retrieve values from the EventPipe and display them in the listbox.

   ```csharp
   _PIEventObject = cdtEvents.Take()
   _pv = _PIEventObject.EventData
   ```

---

**Part E - Test the Application**

1. To test this, first click the Subscribe button. This creates the EventPipe to track new values to CDT158.

2. Next, write new values to the tag CDT158. Go to PI SMT 3.x and choose the Archive Editor in the Data section. (Start > All Programs > PI System > PI System Management Tools.) Use the Archive Editor to find the tag CDT158 and retrieve the recent values. Add a value at the end of the list of values (such as 111). In the Event Time column enter "*". Click the Save button to save it in PI. Try adding a sequence of values such as 111, 222, 333 etc. that you will recognize when retrieved by the application. As you take values from the queue use this to add new values.
Note: random values for CDT158 will also appear occasionally, written by the Random interface that ships with all PI systems.

<table>
<thead>
<tr>
<th>Value</th>
<th>Event Time</th>
<th>Questionable</th>
<th>Annotated</th>
<th>Substituted</th>
</tr>
</thead>
<tbody>
<tr>
<td>157.1411</td>
<td>7/19/2007 7:23:16 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>171.9468</td>
<td>7/19/2007 7:34:16 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>167.041</td>
<td>7/19/2007 7:35:16 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>222</td>
<td>7/19/2007 7:39:25 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>155.0136</td>
<td>7/19/2007 7:39:46 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>250</td>
<td>7/19/2007 7:37:16 PM</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

3. Take one value at a time from the queue by clicking the "Take" button.
4. Take all the values (leaving the queue empty) by clicking the "Take All" button.
5. Test the event handler by clicking the "Update" button. Now, as you write a new value to the PI server it should appear in your application.