PI ProcessBook Add-ins and PI ActiveView, Making them work together

January 2009

OSIsoft, Inc.

777 Davis St., Suite 250
San Leandro, CA 94577 USA
(01) 510-297-5800 (main phone)
(01) 510-357-8136 (fax)
(01) 510-297-5828 (support phone)

support@osisoft.com

Houston, TX
Johnson City, TN
Mayfield Heights, OH
Phoenix, AZ
Savannah, GA
Seattle, WA
Yardley, PA

OSIsoft Australia

Perth, Australia
Auckland, New Zealand

OSI Software GmbH

Altenstadt, Germany

OSI Software Asia Pte Ltd.

Singapore

OSIsoft Canada ULC

Montreal, Canada

OSIsoft, Inc. Representative Office

Shanghai, People’s Republic of China

OSIsoft Japan KK

Tokyo, Japan

OSIsoft Mexico S. De R.L. De C.V.

Mexico City, Mexico

OSIsoft do Brasil Sistemas Ltda.

Sao Paulo, Brazil

Sales Outlets/Distributors

Middle East/North Africa
Republic of South Africa
Russia/Central Asia

South America/Caribbean
Southeast Asia
South Korea Taiwan

www.osisoft.com

Table of Contents

Introduction

Often add-ins written for PI ProcessBook throw errors or disable themselves if PI ActiveView is installed on the same machine.

This is a guide for writing PI ProcessBook add-ins that run correctly in both PI ProcessBook and PI ActiveView. In addition, this guide details how to develop PI ActiveView-only add-ins.

Intended Audience

This document is intended for developers who write or maintain PI ProcessBook add-ins that must also operate correctly, or at least not interfere with, PI ActiveView.

The Root of the Compatibility Problem

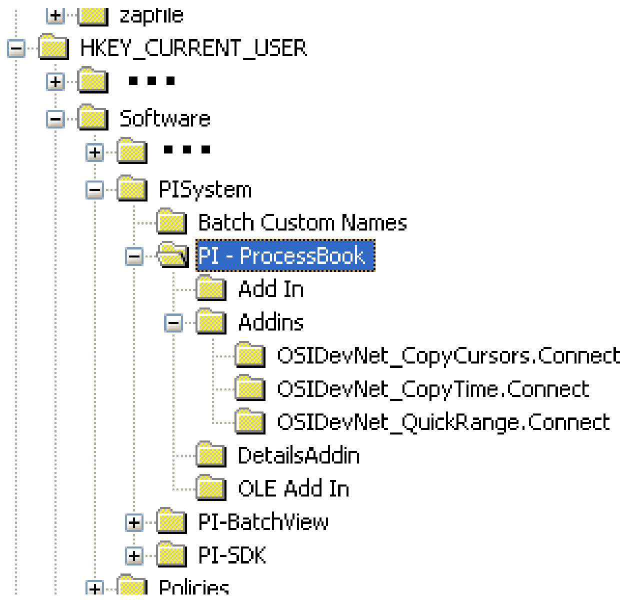
PI ProcessBook and PI ActiveView (versions 2.x and later) share a number of structures that result in any add-in developed for PI ProcessBook and set to load and execute on startup, to also loading with PI ActiveView startup. However, there are functional differences between the two applications of which add-in developers should be aware:

1. PI ActiveView is a read-only application and *build* mode operations are not available
2. PI ActiveView uses an ActiveX control as its user interface and does not support the MDI (Multiple Document Interface) of PI ProcessBook, including menu bars and toolbars.
3. PI ActiveView only works with displays (PDI's) and not workbooks (PIW's)
4. The PI ActiveView Toolbar and Context menu are not the same as PI ProcessBook's and cannot be referenced by an add-in.
5. PI ActiveView is often used with Microsoft Internet Explorer. As such, PDI files are typically loaded from the Microsoft Internet Explorer cache and relative references to other file system objects may not resolve correctly (the PI ProcessBook Button symbol has special code to locate its targets).

6. As with the above: the use, in VBA, of the `ThisDisplay.Path` property points to the cache for PI ActiveView and cannot be used to construct the path to a target object.

Shared Registry Entries

Both applications use the following registry structure to define and load add-ins. The example below shows entries for the CopyCursors, CopyTime and QuickRange add-ins, for example.



```
zaphile
HKEY_CURRENT_USER
  ▪▪▪
  Software
    ▪▪▪
    PISystem
      Batch Custom Names
      PI - ProcessBook
        Add In
        Addins
          OSIDevNet_CopyCursors.Connect
          OSIDevNet_CopyTime.Connect
          OSIDevNet_QuickRange.Connect
        DetailsAddin
        OLE Add In
      PI-BatchView
      PI-SDK
    Policies
```

Moving Forward

While current PI ActiveView versions (up to 3.1.x) throw an error when referencing PI ProcessBook objects like windows and command bars, there is a continuing effort to produce PI ActiveView implementations of them. If proper error handling is used in add-ins, then when and if this occurs then the add-in will not require any updates.

Add-in developers should consider the following:

Does the add-in only work in Build Mode?	If so, then a simple `Exit If Not ProcessBook` test would be sufficient
Would the add-in provide value to PI ActiveView users? (Most Runtime add-ins would.)	If so, then see the discussion and examples in **Problem Workaround**. With good error checking your add-in will 1) not cause errors when PI ActiveView is loaded and 2) function in PI ActiveView when PI ActiveView adds support for additional PI ProcessBook objects and resources. If not, then the `Exit If Not ProcessBook` test would be sufficient

Problem Workaround

Caution must be used when referencing PI ProcessBook objects that do not exist in PI ActiveView. These include but are not limited to:

- CommandBars
- DockingWindows

While these objects are not supported (as of version 3.1), future PI ActiveView releases may in fact support these objects in some fashion (using the existing methods and properties). For add-in developers, the task is to gracefully handle errors that occur when these objects are not available.

Of particular concern is the code executed in the **AddinInstance_OnConnection** and **AddinInstance_OnDisconnection** subroutines. Problems may also be introduced when the add-in captures events (`Dim WithEvents oDisplay as display`) that may also reference un-supported PI ProcessBook objects.

Un-Handled Exceptions

All add-ins should implement exception handlers to prevent errors from bubbling up into PI ProcessBook or PI ActiveView that could result in the add-in being unloaded.

Because PI ActiveView does not implement all PI ProcessBook objects, this is the likely reason for an exception to be thrown. No attempt is made to enumerate these objects in this paper or elsewhere, since good coding practice dictates that *all* external references should have an exception handler as discussed in the following sections.

OnConnection Problem

Typical OnConnection code (such as the VB6 example below), throws an error in ActiveView:

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
    ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
    ByVal AddInInst As Object, custom() As Variant)

    On Error GoTo error_handler

    'Add a reference to the ProcessBook Application
    Set pbApplication = Application

    On Error Resume Next
    Set cbOSIDN = pbApplication.CommandBars("OSIDN-Toolpack")
    On Error GoTo error_handler

    'If it does not exist, create the toolbar.
    If cbOSIDN Is Nothing Then

        Set cbOSIDN = pbApplication.CommandBars.Add("OSIDN-Toolpack", 1, False, False)

            . . .

    Exit Sub

error_handler:
       MsgBox Err.Description
End Sub
```

In ActiveView the line `Set cbOSIDN = pbApplication.CommandBars.Add("OSIDN-Toolpack", 1, False, False)"` throws an Error because the CommandBars object is not implemented and a generic error message is shown to the user whenever *acview.exe* starts. If no error_handler is provided, then the add-in is deactivated.

The following example shows a C# version:

```
public void OnConnection(object Application, Extensibility.ext_ConnectMode ConnectMode,
object AddInInst, ref System.Array custom)
{
        PBObjLib.Application pbApplication;
        PBObjLib.PBCommandBar cbOSIDN;
        PBObjLib.PBCommandBarComboBox cbbQuickRange;
        try
        {

            // Add a reference to the ProcessBook Application
            pbApplication = Application as PBObjLib.Application;
            // Add-in only works in ProcessBook, exit safely here.
            if (pbApplication.Name.ToUpper().CompareTo("PROCBOOK") != 0)
                return;
            // If it doesn't already exist, create a new toolbar for DevNet Add-Ins
            // Check to see if the toolbar exists
            cbOSIDN = pbApplication.CommandBars.Item("OSIDN-Toolpack");

            // If it does not exist, create the toolbar.
            if (cbOSIDN == null)
            {
                cbOSIDN = pbApplication.CommandBars.Add("OSIDN-Toolpack", 1, false, false);

                // Make the toolbar the same height as the first toolbar
                cbOSIDN.Height = pbApplication.CommandBars.Item(1).Height;
            }

            // Set the location of the toolbar
            DefineToolbarLocation();

            // If it doesn't already exist, create a new toolbar combo box for this Add-In
            // Check to see if the toolbar combo box exists
            cbbQuickRange = cbOSIDN.Controls.Item("Quick Range") as PBCommandBarComboBox;

            // If the combo box does not exist, create it.
            if (cbbQuickRange == null)
            {
                cbbQuickRange =
cbOSIDN.Controls.Add(PBObjLib.pbControlType.pbControlComboBox, null, null, null, false) as
PBCommandBarComboBox;
            }
            // Set the combo box's text and tooltip and style
            cbbQuickRange.Caption = "Quick Range";
            cbbQuickRange.ToolTipText = "Quick Time Range Change";
            cbbQuickRange.Style = PBObjLib.pbComboStyle.pbComboNormal;

            // . . .
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

OnConnection Fix

The following code runs without error **AND** allows the add-in to run if PI ActiveView were to implement a CommandBars object:

```
error_handler:
    ' only ProcessBook supports the full object model (CommandBars)
    If LCase(Application.Name) <> "procbook" Then Exit Sub

    ' is ProcessBook error unexpected
    MsgBox Err.Description, vbExclamation, "Quick Range"


End Sub
```

OnDisconnection Problem

OnDisconnect presents a particular problem because of the way PI ActiveView terminates, via an inactivity timer, which invalidates the Application object. Thus a test on the `application.name` fails and results in the add-in being deactivated.

The following OnDisconnection code (as seen in this VB6 example), reports an error in PI ActiveView:

```
Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As
AddInDesignerObjects.ext_DisconnectMode, custom() As Variant)

    On Error GoTo error_handler

    'Remove the Quick Range combo box from the toolbar
        'Verify that the combo box is on the toolbar
        On Error Resume Next
        Set cbbQuickRange = cbOSIDN.Controls.Item("Quick Range")
        On Error GoTo error_handler

        'If the combo box is on the toolbar, delete it.
        If Not (cbbQuickRange Is Nothing) Then
            Set cbbQuickRange = Nothing
            cbOSIDN.Controls.Item("Quick Range").Delete
        End If

    'Write location of the toolbar into the ini file
    Call StoreToolbarLocation

    'Remove toolbar only if there are no controls on it
    If cbOSIDN.Controls.Count = 0 Then cbOSIDN.Delete

     Exit Sub

error_handler:

    MsgBox Err.Description

End Sub
```

OnDisconnection Fix

The following code runs without error:

```
Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As
AddInDesignerObjects.ext_DisconnectMode, custom() As Variant)

    On Error GoTo error_handler

    'Remove the Quick Range combo box from the toolbar
        'Verify that the combo box is on the toolbar
        On Error Resume Next
        Set cbbQuickRange = cbOSIDN.Controls.Item("Quick Range")
        If cbbQuickRange Is Nothing Then Exit Sub ' nothing to do

        On Error GoTo error_handler

        'If the combo box is on the toolbar, delete it.
        If Not (cbbQuickRange Is Nothing) Then
            Set cbbQuickRange = Nothing
            cbOSIDN.Controls.Item("Quick Range").Delete
        End If

    'Write location of the toolbar into the ini file
    Call StoreToolbarLocation

    'Remove toolbar only if there are no controls on it
    If cbOSIDN.Controls.Count = 0 Then cbOSIDN.Delete

     Exit Sub

error_handler:

    MsgBox Err.Description, vbExclamation, "Quick Range"

End Sub
```

RunMode and Event Handlers

Care must be taken to anticipate and handle errors when setting RunMode to False (Build Mode) and when responding to events (VB6: *WithEvents* on the Display object).

In most cases the code fragment

```
If LCase(Application.Name) <> "procbook" Then Exit Sub
```

is sufficient when added to the beginning of an event handler.

.NET Add-ins

.NET add-ins create a distribution problem if the add-in is to work with both PI ProcessBook and PI ActiveView that COM add-ins did not have.

The issue is that the location of the assembly is not specified in the registry. The file must either be present in a location that the framework probes for assemblies to load (e.g., in the same directory as the calling application, a folder specified in the registry as containing public assemblies, or the GAC) or it must contain a codebase registry key under its CLSID key that specifies where the dll is installed. This can be done using `Regasm` with the /codebase switch and should only be done on signed assemblies.

```
Regasm assemblyname /codebase
```

Big Picture

Due to the common architecture of PI ProcessBook and PI ActiveView, all add-ins always run their OnConnect and OnDisconnet routines and provisions need to be made to fully handle errors in both PI ProcessBook and PI ActiveView runtime environments. In many cases, since these errors are caused by the resources needed to activate the add-in (e.g. a CommandBar button), no further special handling is required in the implementation code (that responds to the button click event). However, events fired by PI ProcessBook objects that are also passed to the add-in also provide an avenue to failure when used in PI ActiveView to control other PI ProcessBook objects.Again, the use of error handlers for add-ins can result in add-ins that are well behaved in both applications.

Document History

Andre Downey	Created	January 2009
LDieffenbach	Edited	January 2009