AgileExtender Framework Guide

AgilePoint BPMS v5.0 SP2

Document Revision r5.1.6

November 2011

Contents

Preface

Disclaimer of Warranty

Copyright

Trademarks

Government Rights Legend

Virus-free software policy

AgilePoint recognizes that viruses are a significant security consideration for our customers. To date, we have had no report of AgilePoint BPMS carries any virus. AgilePoint takes the following measures to ensure our software is free of viruses upon delivery:

- AgilePoint is built on top of Microsoft .NET framework. The pre-compiled executable is a.NET Common Language Runtime (CLR) application, not a native machine binary. As far as is known at this time, there are no viruses that infect .NET CLR executables.
- The virtual environment for the product packaging process in is fully isolated and protected, and anti-virus software is installed and running during packaging.
- The deliverable package is scanned by anti-virus software before upload to our customer download site.

Document Revision Numbers

AgilePoint documentation uses the revision number format **r*X.Y.Z***. The letters and numbers in this revision number can be interpreted as follows:

- **r** - Indicates "revision." This helps to differentiate the document *version* numbers, which start with **v**.
- ***X*** - The major version number for AgilePoint BPMS to which this document refers. For example, AgilePoint releases 5.0, 5.0 SP1, and 5.5 would all have an ***X*** value of **5**.
- ***Y*** - The major document revision number. This number typically changes only when either there is a new AgilePoint release, or there are major changes to the document.
- ***Z*** - The minor document revision number. This number is incremented each time the document is republished.

AgilePoint Documentation in PDF and HTML

AgilePoint documentation is provided in both print-friendly (PDF) and web-based (HTML) formats.

Advantages of HTML Documentation

- HTML is the **primary delivery format** for AgilePoint documentation.
- Unified, global **search** across all documentation. PDF documents allow you to search only within the context of a given PDF file.
- **All hyperlinks supported**. Links in PDFs are only supported in certain contexts.
- "One-stop shopping" for all information related to AgilePoint BPMS.
- The HTML documentation is updated more frequently than the PDF documentation. Web-based documentation is updated periodically between AgilePoint releases to address errors and omissions, but the PDF documentation is updated only at the time of a software release.

Advantages of PDF Documentation

PDFs can be more easily **printed**, **archived**, and **transferred** (such as by FTP or email) than HTML documentation.

For more information, see Downloading Files and Sharing Links from the Documentation Library on the AgilePoint Support Portal.

Contacting AgilePoint Sales

AgilePoint is a leading Business Process Management System (BPMS) provider created by a team of driven people who strive to incorporate the principles of relentless innovation for the benefit of our customers. Our mission is to help companies of any size attain and sustain operational success through process excellence.

Headquarters: AgilePoint Corporation 1916C Old Middlefield Way Mountain View, CA 94043, USA

Tel: (650) 968 - 6789

Fax: (650) 968 - 6785

Email: info@agilepoint.com

Web site: www.agilepoint.com

International: For AgilePoint EMEA and AgilePoint Asia Pacific, please call the AgilePoint Corporate Office for contact information.

Contacting Customer Support

To contact AgilePoint Support, please submit a ticket on the AgilePoint Support Portal: http://support.agilepoint.com/SupportPortal/

If you do not have a Support Portal account, you can send an email to request one: support@agilepoint.com

AgileExtender Framework

This document provides information and examples regarding the AgileExtender Framework.

An AgileExtender is a type of AgileShape that allows users to overlay the AgilePoint process model with custom behavior. An AgileExtender is a kind of "meta shape" that runs on top of or in parallel to the process, rather than running inline within the process flow. This provides increased extensibility and allows layers of powerful functionality to be added to the process without making the core process model unnecessarily complex.

You can create custom AgileExtenders using the AgileExtender Framework. The AgileExtender Framework provides a module that captures process-level events and activity-level events. It can also use APIs to interact with the AgilePoint Server engine.

Main Components

AgileExtender is composed of three basic components:

- WFProcessPluggableAdapter
- WFProcessPluggableAdapterDescriptor
- A windows Configuration form that enables the end user to customize the behavior.

WFProcessPluggableAdapterDescriptor and the configuration forms are used at design time, whereas the WFProcessPluggableAdapter is used at runtime.

There are three main parts of an AgileExtender:

- Design time properties
- Deployment time behavior
- Runtime behavior

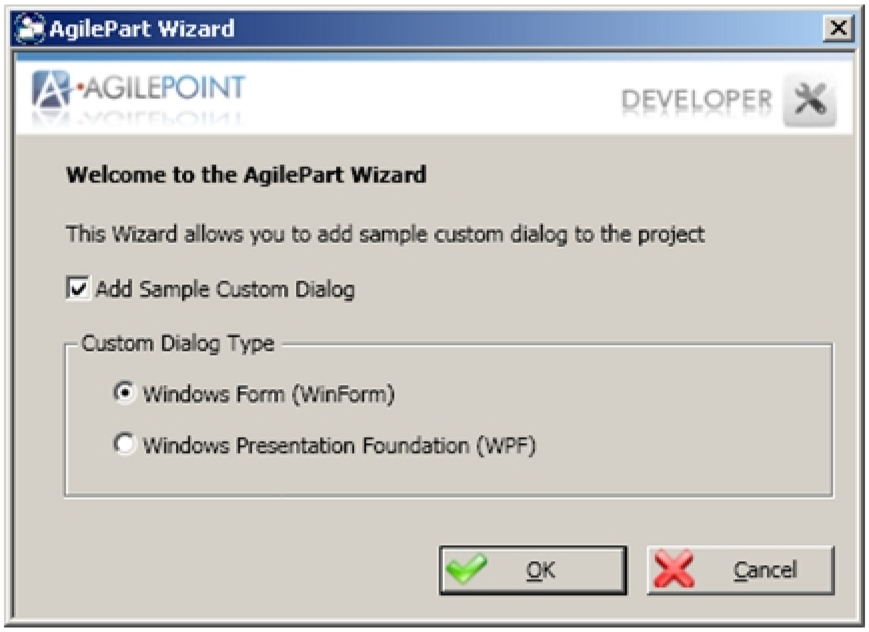
Creating a New AgileExtender Project

To create a new AgileExtender project:

1. In **AgilePoint Developer**, click **File > New > Project**.
2. On the New Project window, click **AgileExtender**.
3. Enter the **Name**, **Location**, and **Solution Name** as required, and click **OK**.
4. On the **Add Sample Custom Dialog** window, Complete the fields as required. For more information, see Add Sample Custom Dialog Window on the AgilePoint Support Portal.

Add Sample Custom Dialog Window

Adds a sample dialog box to your project, which you can use as a template. Adding this dialog can save you time in developing an interface for user input.


AgilePart Wizard

Welcome to the AgilePart Wizard

This Wizard allows you to add sample custom dialog to the project

[✓] Add Sample Custom Dialog

Custom Dialog Type

(●) Windows Form (WinForm)

( ) Windows Presentation Foundation (WPF)

Field Definitions

Field Name	Definition
Add Sample Custom Dialog	If selected, creates a sample dialog box you can use as a template.

Field Name	Definition
Custom Dialog Type	• **Windows Form (WinForm)** - Creates a WinForm type sample dialog box. • **Windows Presentation Foundation (WPF)** - Creates a WPF type sample dialog box.

Generated Files

Once you have created the AgileExtender project, the following files appear in the **Solution Explorer** (in addition to other standard AgilePoint project files):

- **DesignTime > AgileExtender.Design.cs** - Defines the design time properties for the AgileExtender.
- **DesignTime > AgileExtenderConfigForm.cs** - If you chose to add a sample custom dialog, provides a form template you can use to provide users design time input.
- **DeployTime > AgileExtender.Deploy.cs** - Defines the deployment time behavior for the AgileExtender.
- **Runtime > AgileExtender.cs** - Defines the runtime behavior for the AgileExtender.

Design Time Properties

The design time portion of the AgileExtender is used to configure properties that can be used at deployment time or runtime. The Designtime class is inherited from WFProcessPluggableAdapterDescriptor.

Every property has a Category, Description, and Visibility attribute. Multiple properties can be configured under the same Category. The attribute Browsable (true/false) is used to decide the visibility of the property.

You can override the Validate method to write custom logic for validating the user configurations.

The following code sample shows how you can write the design time AgileExtender.

```
using System;
using System.IO;
using System.Xml;
using System.ComponentModel;
using System.Drawing.Design;
using Ascentn.Workflow.Base;
using System.Xml.Serialization;

namespace Ascentn.AgileExtender.Sample
{
    /// <summary>
    /// This class is AgileExtender design time class that
 will be
    /// used by AgilePoint Envision.
    /// </summary>
    public partial class MyAgileExtenderDescriptor :
        WFProcessPluggableAdapterDescriptor
    {
        #region Fields
        private string m_Property1;
        private int m_Property2;
        private string m_ConfigItem1;
        private string m_ConfigItem2;
        #endregion Fields

        #region Constructor

        public MyAgileExtenderDescriptor()
        {
            m_Property1 = string.Empty;
            m_Property2 = 0;
            m_ConfigItem1 = string.Empty;
            m_ConfigItem2 = string.Empty;
        }


        #endregion
        #region [ Exposed Properties ]

        /// <summary>
        /// A sample property of type string
        /// </summary>
        [
            Category("Custom Property"),
            Description("description for Property1..."),
            XmlElement("Property1")
        ]
        public string Property1
        {
            get
            {
                return m_Property1;
            }
            set
            {
                m_Property1 = value;
            }
        }

        /// <summary>
        /// A sample property of type Integer
        /// </summary>
        [
        Category("Custom Property"),
        Description("description for Property2..."),
        XmlElement("Property2")
        ]
        public int Property2
        {
            get
            {
                return m_Property2;
            }
            set
            {
                m_Property2 = value;
            }
        }
        /// <summary>
        /// A sample property of type DialogBox
        /// </summary>

        [
            Category("Custom Property"),
            Description("Configure properties"),
            XmlElement("Configure"),

Editor(typeof(WFProcessPluggableAdapterPropertyModalEditor
               <MyAgileExtenderDescriptor,
AgileExtenderConfigForm>),
               typeof(UITypeEditor))
        ]
        public string Configure
        {
            get
            {
                return "{Sample Configuration}";
            }
        }

        #endregion

        #region [ Hidden Properties ]

        [Browsable(false), XmlIgnore()]
        public override System.Xml.XmlDocument Configuration
        {
            get
            {
                string xml = ShUtil.Serialize(this);
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(xml);
                return xmlDoc;
            }
            set
            {
                XmlDocument xmlDoc = value;
                MyAgileExtenderDescriptor aeDescriptor =

(MyAgileExtenderDescriptor)ShUtil.Deserialize
                    (value.OuterXml, this.GetType());
                this.m_Property1 = aeDescriptor.m_Property1;
                this.m_Property2 = aeDescriptor.m_Property2;
                this.m_ConfigItem1 =
aeDescriptor.m_ConfigItem1;
                this.m_ConfigItem2 =
aeDescriptor.m_ConfigItem2;
            }
        }

        /// <summary>
        /// A sample property for Configuration DialogBox
        /// </summary>
        [
        Category("Custom Property"),
        Description("description for ConfigItem1..."),
        Browsable(false)
        ]
        public string ConfigItem1
        {
            get
            {
                string property = m_ConfigItem1;
                if (property == null) property = "";
                return property;
            }
            set
            {
                m_ConfigItem1 = value;
            }
        }

        /// <summary>
        /// A sample property for Configuration DialogBox
        /// </summary>
        [
        Category("Custom Property"),
        Description("description for ConfigItem2..."),
        Browsable(false)
        ]
        public string ConfigItem2
        {
            get
            {
                string property = m_ConfigItem2;
                if (property == null) property = "";
                return property;
            }
            set
            {
                m_ConfigItem2 = value;
            }
        }

        #endregion

        #region [ Validation methods ]

        /// <summary>
        /// Data Validation
        /// </summary>
        public override void Validate()
        {
            if (string.IsNullOrEmpty(m_Property1))
            {
                throw new InvalidDataException("Property1 of
 Agile
                    Extender is required.");
            }
        }

        #endregion


    }
}
```

Deployment Time Behavior

Deployment Time behavior is useful when you want to perform a custom action during process deployment to AgilePoint Server. Deployment Time behavior runs on AgilePoint Server when process templates are created, checked in, checked out, released, or deleted.

```
using System;
using System.IO;
using System.Xml;
using System.ComponentModel;
using System.Drawing.Design;
using Ascentn.Workflow.Base;
using System.Xml.Serialization;
namespace Ascentn.AgileExtender.Sample
{
    public partial class MyAgileExtenderDescriptor
    {
        #region [ Constructor ]

        public MyAgileExtenderDescriptor(bool designTime)
            : base(designTime)
        {
            if (!base.DesignTime)
            {
                this.CreateProcessDefinition += new

 EventHandler<WFCreateProcessDefinitionArgs>
                    (OnCreateProcessDefnition);
                this.CheckoutProcessDefinition += new

 EventHandler<WFCheckoutProcessDefinitionArgs>
                    (OnCheckoutProcessDefinition);
                this.DeleteProcessDefinition += new

 EventHandler<WFDeleteProcessDefinitionArgs>
                    (OnDeleteProcessDefinition);
                //Add the necessary event handlers
            }
        }

        #endregion

        #region [ Event Handlers ]

        private void OnDeleteProcessDefinition(object sender,
            WFDeleteProcessDefinitionArgs e)
        {
            Logger.WriteLine

 ("MyAgileExtenderDescriptor.OnDeleteProcessDefinition,
                {0}", base.ComponentTypeID);
        }
        private void OnCreateProcessDefnition(object sender,
            WFCreateProcessDefinitionArgs e)
        {
            Logger.WriteLine
 ("MyAgileExtenderDescriptor.OnCreateProcessDefnition,
                {0}", base.ComponentTypeID);
        }

        private void OnCheckoutProcessDefinition(object
 sender,
            WFCheckoutProcessDefinitionArgs e)
        {
            Logger.WriteLine
 ("MyAgileExtenderDescriptor.OnCheckoutProcessDefinition,
                {0}", base.ComponentTypeID);
        }
        #endregion
    }
}
```

The user can get IWFAPI & IWFAdm object from the event args. These objects can then be further used to interact with AgilePoint Server. For deployment time behavior, create a delegate for the deployment events in the constructor, and add code in event handlers to handle the events in a customized way. The following events can be customized according to user requirements:

- **Delete Process Definition** - This event will be triggered when an existing Process definition is deleted from AgilePoint Server.
- **Check In Process Definition** - This event will be triggered when a process is checked in to AgilePoint Server.
- **Check Out Process Definition** - This event is triggered when a process is checked out from AgilePoint Server.
- **Release Process Definition** - This event is triggered by AgilePoint server when a process is released.
- **Disable Process Definition** - This event is triggered by AgilePoint Server when a process is disabled by AgilePoint server.

Case study

Problem

Whenever a new AgileForm based process is created or checked in to AgilePoint Server from either Envision or Enterprise Manager, the form definition must be inserted in a custom database.

Solution

Create an AgileExtender, and write event handlers for create and check in process definition events.

In the event handlers, get the form definition, and insert in the custom database. These events (create process and check in process) will be triggered by AgilePoint Server whenever a new AgileForm based process is created or checked in to AgilePoint Server.

```
public AgileFormAgileExtenderDescriptor(bool designTime)
    : base(designTime)
{
    if (!base.DesignTime)
    {
        this.CheckinProcessDefinition+=new
            EventHandler<WFCheckinProcessDefinitionArgs>
            (OnCheckInProcessDefination);
        this.CreateProcessDefnition+=new
            EventHandler<WFCreateProcessDefinitionArgs>

 (AgileFormAgileExtenderDescriptor_CreateProcessDefnition);
    }
}
```

Runtime Behavior

Runtime behavior occurs on the AgilePoint Server as the process model is running. The runtime class is inherited from WFProcessPluggableAdapter. The various events that can be handled via runtime AgileExtender are listed below:

- **Assign Work Item** - Triggered when a work item is assigned.
- **Cancel Work Item** - Triggered when a work item is canceled.
- **Complete Work Item** - Triggered when a work item is completed.
- **Enter Activity Instance** - Triggered when the process enters an activity instance.
- **Leave Activity Instance** - Triggered when the process leaves an activity instance.
- **Reassign Work Item** - Triggered when an activity is reassigned.
- **Work Item Overdue** - Triggered when the assigned work item goes overdue.
- **Work Item Assigned** - Triggered when an work item is assigned.
- **Resolving Pool Members** - Triggered while resolving the pool members.
- **Activate Work Item** - Triggered while activating the work item.
- **Cancel Process Instance** - Triggered to cancel a running process instance.
- **Complete Procedure** - Triggered to complete a procedure.
- **Create Process Instance** - Creates a new process instance.
- **Deliver Notification** - Responsible for mail delivery notifications.
- **Process Instance Faulting** - Triggered during process instance faulting.
- **Resume Process Instance**- Resumes a suspended process instance.
- **Suspend Process Instance** - Suspends a running process instance.
- **Stop Process Instance** - Stops a running process instance.
- **Start Process Instance** - Starts a process instance.
- **Scheduled Item Timeout** -Triggered when a scheduled item is timed out.
- **Update Custom Attributes** - Triggered to update custom attributes.
- **Rollback Process Instance** - Rolls back a process instance.
- **Migrate Process Instance** - Responsible for migrating a process instance.
- **Query Custom Attributes** - Triggered to query custom attributes.
- **CompletingWorkItem -** Triggered before the completion of work item is committed.
- **CancellingWorkItem -** Triggered before the cancellation of work item is committed.

The Typical Constructor will look as follows :

```
using System;
using System.ComponentModel;
using System.Collections;
using Ascentn.Workflow.Base;

namespace Ascentn.AgileExtender.Sample
{
    /// <summary>
    /// This class is AgileExtender runtime class that will
 be
    /// invoked by AgilePoint Server.
    /// </summary>
    [AgileExtender("{305F5DC5-14F3-4151-946E-2D594DA695E5}",
        "AgileExtender Event",
 typeof(MyAgileExtenderDescriptor))]

    public class MyAgileExtender : WFProcessPluggableAdapter
    {
        #region Constructor

        public MyAgileExtender(WFProcessInstance instance)
            : base(instance)
        {
            base.AssignWorkItem += new
                EventHandler(OnAssignWorkItem);
            base.CancelWorkItem += new
                EventHandler(OnCancelWorkItem);
            base.CompleteWorkItem += new
                EventHandler(OnCompleteWorkItem);
            base.EnterActivityInstance += new
                EventHandler(OnEnterActivityInstance);
            base.LeaveActivityInstance += new
                EventHandler(OnLeaveActivityInstance);
            base.ReassignWorkItem += new
                EventHandler(OnReassignWorkItem);
            base.WorkItemOverdue += new
                EventHandler(OnWorkItemOverdue);
            base.WorkItemAssigned += new
                EventHandler(OnWorkItemAssigned);
            base.ResolvingPoolMembers += new
                EventHandler(OnResolvingPoolMembers);

            base.CancelProcessInstance += new
                EventHandler(OnCancelProcessInstance);
            base.CompleteProcedure += new
                EventHandler(OnCompleteProcedure);
            base.CreateProcessInstance += new
                EventHandler(OnCreateProcessInstance);
            base.DeliverNotification += new
                EventHandler(OnDeliverNotification);
            base.ProcessInstanceFaulting += new
                EventHandler(OnProcessInstanceFaulting);
            base.ResumeProcessInstance += new
                EventHandler(OnResumeProcessInstance);
            base.SuspendProcessInstance += new
                EventHandler(OnSuspendProcessInstance);
            base.StopProcessInstance += new
                EventHandler(OnStopProcessInstance);
            base.StartProcessInstance += new
                EventHandler(OnStartProcessInstance);
            base.ScheduledItemTimeout += new
                EventHandler(OnScheduledItemTimeout);
            base.UpdateCustomAttributes += new
                EventHandler(OnUpdateCustomAttributes);
            base.RollbackProcessInstance += new
                EventHandler(OnRollbackProcessInstance);
            base.MigrateProcessInstance += new
                EventHandler(OnMigrateProcessInstance);
            base.CompletingWorkItem += new
                EventHandler(OnCompletingWorkItem);
            base.CancallingWorkItem += new
                EventHandler(OnCancellingWorkItem);
        }

        #endregion

        #region Event Handlers

        private void OnMigrateProcessInstance(Object sender,
            System.EventArgs e)
        {
            WFMigrateProcessInstanceEventArgs args = e as
                WFMigrateProcessInstanceEventArgs;

Logger.WriteLine("MyAgileExtender.OnMigrateProcessInstance,
                {0}", args.MigrationInstruction);
            //Put your code here
        }

        private void OnRollbackProcessInstance(Object sender,
            System.EventArgs e)
        {
            WFRollbackProcessInstanceEventArgs args = e as
                WFRollbackProcessInstanceEventArgs;

            //Put your code here

Logger.WriteLine("MyAgileExtender.OnRollbackProcessInstance,
                target activity instance ID='{0}'",
                args.ActivityInstanceID);
        }

        private void OnUpdateCustomAttributes(Object sender,
            System.EventArgs e)
        {
            WFUpdateCustomAttributesEventArgs args = e as
                WFUpdateCustomAttributesEventArgs;

Logger.WriteLine("MyAgileExtender.OnUpdateCustomAttributes,
                ModifiedDate={0}, ModifiedBy={1}, before={2},
                after={3}", args.SentDate, args.ModifiedBy,
                args.BeforeUpdate.AttrXml,
 args.AfterUpdate.AttrXml);

            //Put your code here
        }

        private void OnScheduledItemTimeout(Object sender,
            System.EventArgs e)
        {
            WFScheduleItemTimeoutEventArgs args = e as
                WFScheduleItemTimeoutEventArgs;

 Logger.WriteLine("MyAgileExtender.OnScheduledItemTimeout,
                ItemID={0}, Parameter={1}", args.ItemID,
                args.Parameter);

            //Put your code here
        }

        private void OnStartProcessInstance(Object sender,
            System.EventArgs e)
        {
            System.Xml.XmlDocument config =
 base.Configuration;

 Logger.WriteLine("MyAgileExtender.OnStartProcessInstance,
                config={0}", config.OuterXml);

            //Put your code here
        }

        private void OnStopProcessInstance(Object sender,
            System.EventArgs e)
        {

 Logger.WriteLine("MyAgileExtender.OnStopProcessInstance");

            //Put your code here
        }

        private void OnSuspendProcessInstance(Object sender,
            System.EventArgs e)
        {

 Logger.WriteLine("MyAgileExtender.OnSuspendProcessInstance");

            //Put your code here
        }

        private void OnResumeProcessInstance(Object sender,
            System.EventArgs e)
        {

 Logger.WriteLine("MyAgileExtender.OnResumeProcessInstance");
```

```
            //Put your code here
        }

        private void OnProcessInstanceFaulting(Object sender,
            System.EventArgs e)
        {
            WFProcessInstanceFaultingEventArgs args = e as
                WFProcessInstanceFaultingEventArgs;

Logger.WriteLine("MyAgileExtender.OnProcessInstanceFaulting,
                {0}", args.FaultingException);

            //Put your code here
        }

        private void OnDeliverNotification(Object sender,
            System.EventArgs e)
        {

Logger.WriteLine("MyAgileExtender.OnDeliverNotification");

            //Put your code here
        }

        private void OnCompleteProcedure(Object sender,
            System.EventArgs e)
        {

Logger.WriteLine("MyAgileExtender.OnCompleteProcedure");
        }

        private void OnCreateProcessInstance(Object sender,
            System.EventArgs e)
        {

Logger.WriteLine("MyAgileExtender.OnCreateProcessInstance");

            //Put your code here
        }

        private void OnCancelProcessInstance(Object sender,
            System.EventArgs e)
        {

Logger.WriteLine("MyAgileExtender.OnCancelProcessInstance");

            //Put your code here
        }

        private void OnResolvingPoolMembers(Object sender,
            System.EventArgs e)
        {
            WFActivityInstanceEventArgs args = e as
                WFActivityInstanceEventArgs;
```

```
Logger.WriteLine("MyAgileExtender.OnResolvingPoolMembers,
               Activity  Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnEnterActivityInstance(Object sender,
           System.EventArgs e)
       {
           WFActivityInstanceEventArgs args = e as
               WFActivityInstanceEventArgs;

Logger.WriteLine("MyAgileExtender.OnEnterActivityInstance,
               Activity Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnAssignWorkItem(Object sender,
           System.EventArgs e)
       {
           WFGenerateManualWorkItemEventArgs args = e as
               WFGenerateManualWorkItemEventArgs;

Logger.WriteLine("MyAgileExtender.OnAssignWorkItem,
               Activity Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnCancelWorkItem(Object sender,
           System.EventArgs e)
       {
           WFManualWorkItemEventArgs args = e as
               WFManualWorkItemEventArgs;

Logger.WriteLine("MyAgileExtender.OnCancelWorkItem,
               Activity Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnCompleteWorkItem(Object sender,
           System.EventArgs e)
       {
           WFManualWorkItemEventArgs args = e as
               WFManualWorkItemEventArgs;

Logger.WriteLine("MyAgileExtender.OnCompleteWorkItem,
               Activity Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnLeaveActivityInstance(Object sender,
           System.EventArgs e)
       {
           WFActivityInstanceEventArgs args = e as
               WFActivityInstanceEventArgs;

Logger.WriteLine("MyAgileExtender.OnLeaveActivityInstance,
               Activity  Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnReassignWorkItem(Object sender,
           System.EventArgs e)
       {
           WFManualWorkItemEventArgs args = e as
               WFManualWorkItemEventArgs;

Logger.WriteLine("MyAgileExtender.OnReassignWorkItem,
               Activity Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
       }

       private void OnWorkItemAssigned(Object sender,
           System.EventArgs e)
       {
           WFManualWorkItemEventArgs args = e as
               WFManualWorkItemEventArgs;

Logger.WriteLine("MyAgileExtender.OnWorkItemAssigned,
               Activity Instance={0}, {1} workitem(s)
assigned",
               args.ActivityInstance, args.WorkItems.Length);

           //Put your code here
       }

       private void OnWorkItemOverdue(Object sender,
           System.EventArgs e)
       {
           WFOverdueWorkItemEventArgs args = e as
               WFOverdueWorkItemEventArgs;
           WFManualWorkItem workItem = args.WorkItem as
               WFManualWorkItem;

Logger.WriteLine("MyAgileExtender.OnWorkItemOverdue,
               Activity Instance={0}, {1} is about to
overdue",
               args.ActivityInstance, workItem.Name);

           //Put your code here
       }

       private void OnCompletingWorkItem (Object sender,
           EventArgs e)
       {
          WFManualWorkItemEventArgs args = e as
               WFManualWorkItemEventArgs;

Logger.WriteLine("AgileExtender.OnCompleteWorkItem,
               Activity  Instance={0}",
               args.ActivityInstance.DisplayName);

          //Put your code here
       }

       private void OnCancellingWorkItem (Object sender,
           EventArgs e)
       {
           WFCancellingManualWorkItemEventArgs args =
               e as WFCancellingManualWorkItemEventArgs;

Logger.WriteLine("AgileExtender.OnCancellingWorkItem,
               Activity  Instance={0}",
               args.ActivityInstance.DisplayName);

           //Put your code here
        }

       #endregion
   }
}
```

Creating and Registering a Shape in Envision

Once you have created the AgileExtender code, you must register it in Envision.

AgilePoint Server

To download and register an AgileExtender that is already deployed in AgilePoint Server:

1. In AgilePoint Envision, click **File > Extend AgilePoint Envision > Register AgileExtender**.
2. Click the **Download** button.
3. Connect to the AgilePoint Server.
4. Select an AgileExtender that is available on the server and click **OK**.

Local File

To register an AgileExtender DLL that is located on your computer's file system:

1. Click the **Add** button.
2. Select the appropriate DLL file and click **Open**.

Once an AgileExtender assembly has been registered, the AgileExtender and its methods will be displayed in the AgileExtender Registration dialog (as shown below) and they can be used in process templates.

Registering an AgileExtender in Envision

This command registers a custom AgileExtender component with AgilePoint Envision. Custom AgileExtender components must be registered before they can be used in a process template. After selecting this option, the registration dialog will be displayed.

1. In Envision, click **File > Extend AgilePoint > Register AgileExtender**.
2. On the **AgileExtender Registration** dialog box, click **Add**.
3. Select the DLL file for the AgileExtender , and click **Open**.

Once an AgileExtender assembly has been registered, the AgileExtender and its methods will be displayed in the AgileExtender Registration dialog, and the methods can be used in process templates.