Workspace Desktop Edition Developer's Guide

Customizing Work Items

4/10/2025

Customizing Work Items

Purpose: To provide information about the customization of WorkItems.
Available since: 8.1.000.21

Contents

Before You Start

Work items are interactions which are not predefined in Interaction Workspace, in opposition to e-mail, instant messaging, voice, and chat interactions. You can define these work items to extend the Interaction Workspace Specificities; for instance, if your solution integrates sms, or fax, or additional media, you can define new work items to handle these interactions. The following step-by-step tutorial explains how to configure your solution to support work items, and then, how to customize your Interaction Workspace to integrate them. The code snippets are extracted from the Genesyslab.Desktop.Modules.CustomWorkItemSample, available in the Interaction Workspace samples.

Set Up System Configuration

First of all, set up the System Configuration as described in Enabling an agent to use Workitems to handle custom media types of the Deployment Guide to create a new Media Type.

Displaying the content of Workitems

To display the content of work item interactions, you need to implement a customized view as explained in Customize Views and Regions. The Genesyslab.Desktop.Modules.CustomWorkItemSample.Window.Content shows how to create a new view handling this task:

- `ICustomWorkItemViewModel.cs` and `ICustomWorkItemView.cs` implement de MVVM pattern.
- `CustomWorkItemViewModel.cs` manages the workitem interaction as `IInteractionOpenMedia`.
- `CustomWorkItemView.xaml` and `CustomWorkItemView.xaml.cs` display the interaction's data and implement the `ICustomWorkItemView` interface, in addition to using the model.

In `CustomWorkItemView.xaml.cs`, the CustomWorkItemView instance is initialized with the model which contains the open media interaction. **[C#]**

```
// file CustomWorkItemView.xaml.cs
public partial class CustomWorkItemView : UserControl, ICustomWorkItemView
{
        public CustomWorkItemView(ICustomWorkItemViewModel customWorkItemViewModel)
        {
                Model = customWorkItemViewModel;
                InitializeComponent();
                //...
        }
//...
}
```

At creation, this class retrieves the open media interaction from the application's context dictionary, as follows: **[C#]**

```
// file CustomWorkItemView.xaml.cs
///IView members
public object Context { get; set; }
public void Create()
{
         Model.Interaction = contextDictionary.TryGetValue("Interaction") as
IInteractionOpenMedia;
}
```

Then, you can bind the properties in your XAML code. For instance, in the CustomWorkItemSample, the content grid displays a label media type and the associated value, defined as follows: **[XAML]**

```
<!-- file CustomWorkItemView.xaml -->
<Label localization:Translate.Uid="Windows.CustomWorkItemView.MediaType"
HorizontalAlignment="Left" VerticalAlignment="Center" Grid.Column="0" Grid.Row="0"
Content="{localization:Translate Default=Media Type:}"/>
<Label HorizontalAlignment="Left" VerticalAlignment="Center" Grid.Column="1" Grid.Row="0"
Content="{Binding Interaction.EntrepriseInteractionCurrent.IdType.SubMediaType}"/>
```

The `CustomWorkItemSampleModule` class implements the `IModule` interface and adds the new customized view to the list of active views, as shown here in the `RegisterViewsAndServices()` method called by the : **[C#]**

```
// file CustomWorkItemSampleModule.cs
protected void RegisterViewsAndServices()
{
        //...
        container.RegisterType<ICustomWorkItemViewModel, CustomWorkItemViewModel>();
        container.RegisterType<ICustomWorkItemView, CustomWorkItemView>();
}
```

Handle the work item

Using Work Items as described in Inserting a Command in a Chain, you can perform the following actions on work items:

- Accept
- Decline
- Mark done
- Attach data
- Search and/or move to workbin
- And so on.

Or, you can use the Open Media Service of the Enterprise SDK API to perform additional actions. In that case, you must create your commands. The `CustomWorkItemToolBarView.xaml` in `Genesyslab.Desktop.Modules.CustomWorkItemSample.Toolbar` implements a stop processing button, by using the `CustomWorkitemStopProcessingCommand` command created in the `Genesyslab.Desktop.Modules.CustomWorkItemSample.CustomWorkitemCommand.cs` file. The creation of this command follows the guidelines of Creating a Command. At first, the sample implements the `IElementOfCommand` interface which retrieves the `Genesyslab.Enterprise.Services.IOpenMediaService`, as shown here: **[C#]**

```
// file CustomWorkitemCommand.cs
class CustomWorkitemCommand : IElementOfCommand
{
   protected readonly IUnityContainer container;
   protected ILogger log;
   protected IOpenMediaService openMediaService;
   protected const int timeout = 10000;
   public CustomWorkitemCommand(IUnityContainer container)
   {
      this.container = container;
      this.log = container.Resolve<ILogger>();
      this.log = log.CreateChildLogger("CustomWorkitemCommand");
      IEnterpriseServiceProvider enterpriseServiceProvider =
container.Resolve<IEnterpriseServiceProvider>();
      this.openMediaService =
enterpriseServiceProvider.Resolve<IOpenMediaService>("openmediaService");
   }
   public virtual string Name { get; set; }
   #region IElementOfCommand Members
   public virtual bool Execute(IDictionary<string, object> parameters, IProgressUpdater
progressUpdater)
   {
      return false;
   }
   #endregion
}
```

The `CustomWorkitemCommand` class is a base class for further specific commands. In this example, it uses the Open Media Service to stop processing the interaction, as shown in the following code snippet. **[C#]**

```
// file CustomWorkitemCommand.cs
class CustomWorkitemStopProcessingCommand : CustomWorkitemCommand
{
   public CustomWorkitemStopProcessingCommand(IUnityContainer container) : base(container) { }
   public override bool Execute(IDictionary<string, object> parameters, IProgressUpdater
progressUpdater)
   {
      log.Info("CustomWorkitemStopProcessingCommand");
      IInteractionOpenMedia interactionOpenMedia =
         parameters.TryGetValue("CommandParameter") as IInteractionOpenMedia;
      try
      {
         if ((interactionOpenMedia != null)
         && (interactionOpenMedia.EntrepriseOpenMediaInteractionCurrent != null))
         {
            if (!interactionOpenMedia.EntrepriseOpenMediaInteractionCurrent.IsInWorkflow)
            {
               return false;
            }

openMediaService.StopProcessing(interactionOpenMedia.EntrepriseOpenMediaInteractionCurrent,
                  parameters.TryGetValue("Reason") as KeyValueCollection,
                  parameters.TryGetValue("Extensions") as KeyValueCollection);
         }
         return false;
      }
      catch (Exception exp)
      {
         log.Error("CustomWorkitemStopProcessingCommand StopProcessing, Exception "
                        + interactionOpenMedia, exp);
         return true;
```

```
      }
   }
}
```

The `Genesyslab.Desktop.Modules.CustomWorkItemSample.CustomWorkItemSampleModule` implements the `IModule` interface and is responsible for adding the new command to the chain of commands. **[C#]**

```
/// file CustomWorkItemSample.cs
void RegisterCommands()
{
   log.Debug("RegisterCommands()");
   ICommandManager commandManager = container.Resolve<ICommandManager>();
   commandManager.AddCommandToChainOfCommand("InteractionCustomWorkitemStopProcessing",
               new List<CommandActivator>() { new CommandActivator() {
                   CommandType = typeof(CustomWorkitemStopProcessingCommand) ,
                   Name="StopProcessing" } });
}
```

In the `CustomWorkItemToolBarView` class, the event-handler of the `Stop Processing` button retrieves and executes the command: **[C#]**

```
// File: CustomWorkItemSample.Windows.ToolBarView.cs
private void StopProcessingButton_Click(object sender, System.Windows.RoutedEventArgs e)
{
   IChainOfCommand Command = container.Resolve<ICommandManager>().
             GetChainOfCommandByName("InteractionCustomWorkitemStopProcessing");
   Utils.ExecuteAsynchronousCommand(Command,
      new Dictionary<string, object>() { { "CommandParameter", Model.Interaction } },
      StopProcessingButton);
}
```

Customizing Workitem Icons

As an additional customization step for new work items, you can create icons to facilitate the identification of the information related to work items. In that purpose, you need to create or edit a dictionary file, such as for instance, the `Genesyslab.Desktop.Modules.CustomWorkItemSample.en-US.xml` xml file in the `Genesyslab.Desktop.Modules.CustomWorkItemSample` project. Declare your new icon as follows:

```
<Value Id="<myworkitemobjectId>.<workitemchannel>" ImageUrl=<ImagePath> IcoUrl=<IcoPath> />
```

- where `myworkitemobjectId` is the object ID customized in the Interaction Workspace.

Name	Object Id	Customization example
Media Channel Icon	Channel.OpenMedia.WorkItem.Image	<Value Id="Channel.OpenMedia.WorkItem.Image.custom ImageUrl="<ImagePath>" />
Interaction Icon	InteractionWorkItem.Image	<Value Id="InteractionWorkItem.Image.customworkitem Ico="<IcoPath>" ImageUrl="<ImagePath>"/>
Transfer Button	Windows.ToolbarWorkitemView.	<Value

Name	Object Id	Customization example
	ButtonTransfer	Id="Windows.ToolbarWorkitemView.ButtonTransf ToolTip="Custom WorkItem Transfer" ImageUrl="<ImagePath>"/>
Transfer Menu	Windows.ToolbarWorkitemView. MenuItemOneStepTransferDialer	<Value Id="Windows.ToolbarWorkitemView.MenuItemOn Text="Custom Workitem _Transfer" Tooltip="Custom Workitem Transfer" ImageUrl="<ImagePath>" />
'Move to Workbin' button	Windows.ToolbarWorkitemView. ButtonMoveToWorkbin	<Value Id="Windows.ToolbarWorkitemView. ButtonMoveToWorkbin.<workitemchannel>"Tool Move to workbin" ImageUrl="<ImagePath>"/>
New Custom Button in WorkItem ToolBar	Windows.ToolbarCustomWorkItemView.<ButtonName>	<Value Id="Windows.ToolbarCustomWorkItemView.Butto ToolTip="Stop Processing" Text="Stop Processing"/>
Interaction icon in the Interaction History	Contacts.ContactHistoryView. InteractionWorkItem	<Value Id="Contacts.ContactHistoryView.InteractionWor ImageUrl="<ImagePath>"/>
Interaction Icon in Workin management	InteractionWorkItem.MediaType.Image	<Value Id="InteractionWorkItem.MediaType.Image.<wor ImageUrl="<ImagePath>"/>

For instance, here is the `Genesyslab.Desktop.Modules.CustomWorkItemSample.en-US.xml` xml file in the `Genesyslab.Desktop.Modules.CustomWorkItemSample` project, which customize label, images, and icons.

[XML]

```
<?xml version="1.0" encoding="utf-8" ?>
<Dictionary EnglishName="English" CultureName="English" Culture="en-US"
CustomDictionary="true">
 <Value Id="Channel.OpenMedia.WorkItem.Image.customworkitem"

ImageUrl="pack://application:,,,/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/
Images/CustomWorkItem.png" />
 <Value Id="Windows.ToolbarCustomWorkItemView.Button1" ToolTip="Stop Processing" Text="Stop
Processing"/>
 <Value Id="Windows.CustomWorkItemView.MediaType" Content="Media Type:"/>
 <Value Id="Windows.CustomWorkItemView.InteractionType" Content="Interaction Type:"/>
 <Value Id="Windows.CustomWorkItemView.InteractionSubType" Content="Interaction Sub-Type:" />
 <Value Id="Windows.CustomWorkItemView.MyCustomContent" Content="My custom content" />
 <Value Id="InteractionWorkItem.Image.customworkitem"
    Ico="pack://application:,,,/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/
Images/CustomWorkItem.ico"
    ImageUrl="/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/Images/
CustomWorkItem.png"/>
 <Value Id="Windows.ToolbarWorkitemView.ButtonTransfer.customworkitem" ToolTip="Custom
WorkItem Transfer"
    ImageUrl="/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/Images/
TransferCustomWorkItem.png"/>
 <Value Id="Windows.ToolbarWorkitemView.ButtonMoveToWorkbin.customworkitem"
    ToolTip="Move To Workbin In Progress"
```

```
    ImageUrl="/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/Images/
CustomMoveToWorkbin.png"/>
 <Value Id="Windows.ToolbarWorkitemView.MenuItemOneStepTransferDialer.customworkitem"
    Text="Custom Workitem _Transfer"
    Tooltip="Custom Workitem Transfer"
    ImageUrl="/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/Images/
TransferCustomWorkItem.png" />
 <Value Id="Contacts.ContactHistoryView.InteractionWorkItem.customworkitem"
     ImageUrl="/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/Images/
MediaTypeCustomWorkItem.png"/>
 <Value Id="InteractionWorkItem.MediaType.Image.customworkitem"
   ImageUrl="pack://application:,,,/Genesyslab.Desktop.Modules.CustomWorkItemSample;component/
Images/MediaTypeCustomWorkItem.png"/>
</Dictionary>
```

Read Next

Use Customizable Commands