WebFront for Service Manager

Authoring Guide

Gridpro AB

Rev: 2.10.7509 (SCSM 2012 versions) & 3.0.7509 (SCSM 2016 & later)

Published: August 2020

Contents

Purpose

The purpose of this document is to provide assistance when customizing forms in WebFront for Service Manager.

Introduction

Starting with version 2.1 of WebFront for Service Manager it is possible to customize forms in WebFront. This enables you to take advantage of your data model extensions, made with the Service Manager Authoring Tool or other method. Form customizations in WebFront for Service Manager is done directly in the browser by a Service Manager Administrator.

Since WebFront utilizes the XAML language to describe form customizations this allows WebFront to be very flexible when it comes to fulfilling specific customer needs.

Limitations

Current release:

- Only forms available out-of-the-box in WebFront can be customized
- The Generic Form which is used to show work items and configuration items that don't have a specific form associated with it cannot be customized

Prerequisites

To be able to customize WebFront forms you need to be:

- Running version 2.1 or higher
- Member of the Service Manager Administrator Role*

** If you are not a member of this role you will not be able to see the customization menus in the WebFront forms*

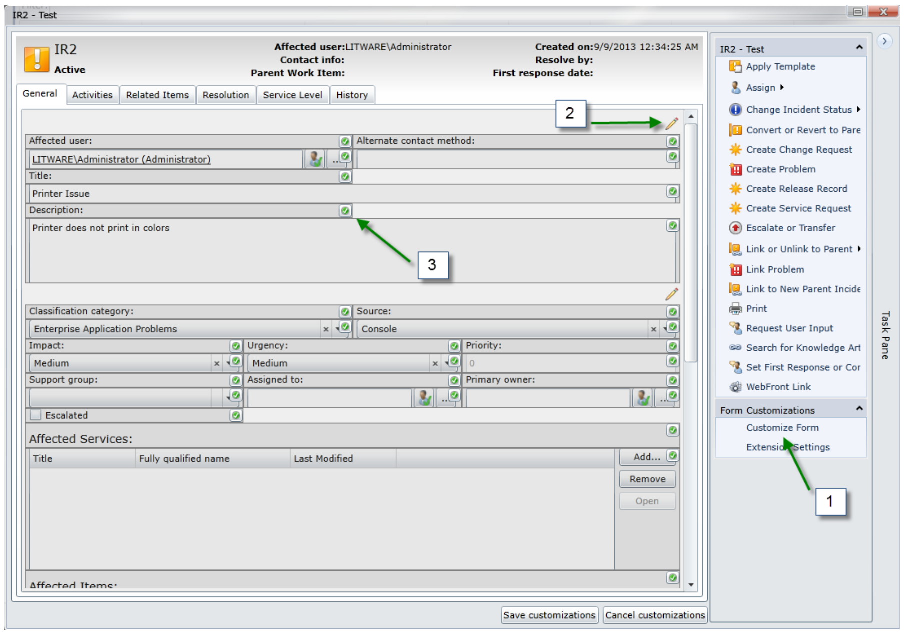
Customizing Forms

Form customization is done directly in the browser when working with WebFront for Service Manager. Each form has different zones that can be customized with your different controls bound to your data model extensions. In the following sections, we will go through how this is accomplished.

Launching the Customization Editor

WebFront has a Customization Editor that can be enabled from within each WebFront form.

To launch the Customization Editor a form in WebFront	
☐	Open WebFront as a Service Manager Administrator
☐	Open the form you want to customize by simply viewing a suitable work item or configuration item associated with the form
☐	In the task menu to the right, click **Customize Form** (Arrow 1 in figure 1)
☐	Click the pencil in the upper right corner of the zone you want to edit to launch the Customization Editor (Arrow 2 in Figure 1)
☐	You can hide controls by clicking green checkbox, this will toggle the control to hidden (Arrow 3 in Figure 1)



IR2 - Test

IR2
Active

Affected user:LITWARE\Administrator
Contact info:
Parent Work Item:

Created on:9/9/2013 12:34:25 AM
Resolve by:
First response date:

Affected user: LITWARE\Administrator (Administrator)
Alternate contact method:
Title: Printer Issue
Description: Printer does not print in colors
Classification category: Enterprise Application Problems
Source: Console
Impact: Medium
Urgency: Medium
Priority: 0
Support group:
Assigned to:
Primary owner:
Escalated

Affected Services:

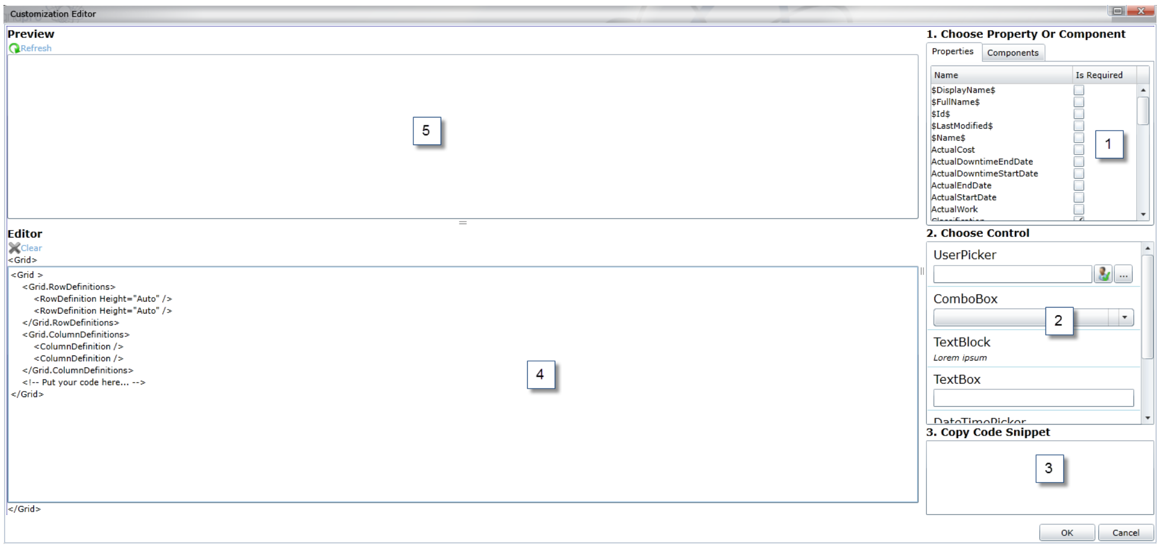
Title	Fully qualified name	Last Modified

Affected Items:

Figure 1. How to launch the Customization Editor. The Form Customization menu is only accessible as a Service Manager Administrator.

Customization Editor

The Customization Editor consists of the following areas.



```
<Grid>
<Grid >
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
   </Grid.ColumnDefinitions>
   <!-- Put your code here... -->
</Grid>
</Grid>
```

Figure 2. Customization Editor – Overview.

Properties/Components (1)

Shows which properties and relationships that are available to the current form, including data model extensions. You can also change if a property or relationship should be required or not. *Note: Properties that are required by the data model in Service Manager cannot be made "not required"*.

Control toolbox (2)

Shows commonly used controls. If a property/component has been selected controls suitable for usage are tagged with a green check mark.

Code snippet (3)

Code snippet for the selected control with a predefined binding to the currently selected property/component.

Editor (4)

Where you specify the XAML code that should be used to render your customization. This area is pre-set with a Grid that has two columns and two rows.

Preview (5)

Used to display a pre-view of your current XAML code and how it will render on the form.

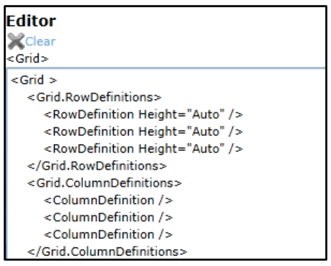
Silverlight XAML Basics

Before starting to customize your forms, it is good to have some basic knowledge on how common controls like the **Grid** and **StackPanel** works in Silverlight XAML. If you do not already know this a good place to start is to get familiar with the Grid control to get to know how to work with rows and columns: http://msdn.microsoft.com/en-us/library/ms610550(v=vs.95).aspx

We have also added some useful information about Columns and Margins below.

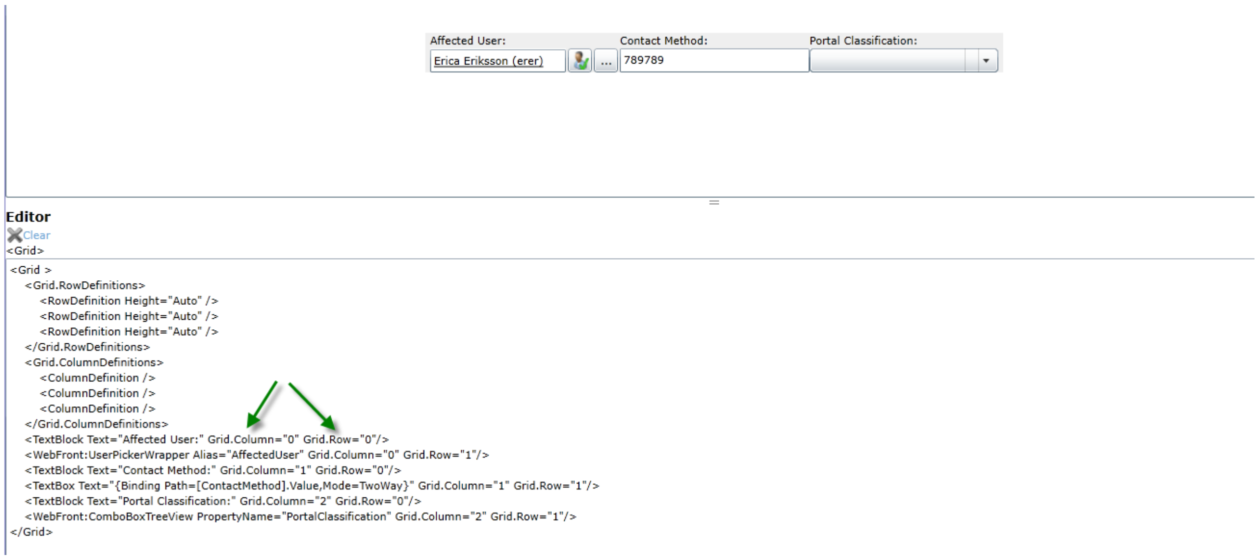
Columns

The grid in the Customization Editor only contain definitions for two columns by default. If you want to have three columns in an edit zone you need to add a RowDefinition in the RowDefinitions section and ColumnDefinition in the ColumnDefinitions section (see picture below).



```
<Grid >
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
   </Grid.ColumnDefinitions>
```

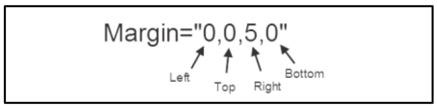
Then you use the Grid.Column and Grid.Row to control the placement column and row (see picture below).

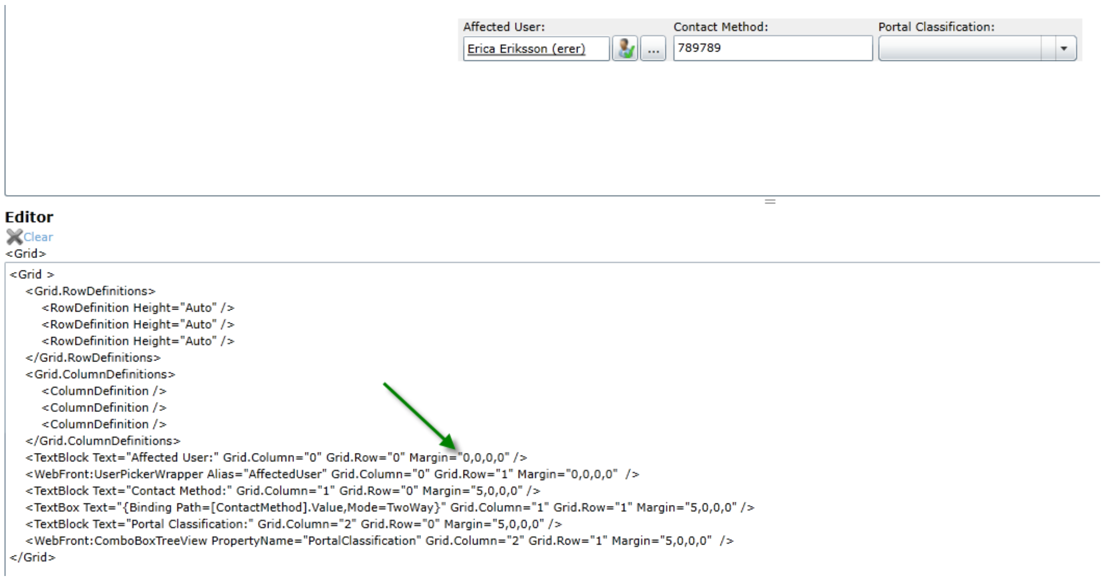


```
<Grid >
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
   </Grid.ColumnDefinitions>
   <TextBlock Text="Affected User:" Grid.Column="0" Grid.Row="0"/>
   <WebFront:UserPickerWrapper Alias="AffectedUser" Grid.Column="0" Grid.Row="1"/>
   <TextBlock Text="Contact Method:" Grid.Column="1" Grid.Row="0"/>
   <TextBox Text="{Binding Path=[ContactMethod].Value,Mode=TwoWay}" Grid.Column="1" Grid.Row="1"/>
   <TextBlock Text="Portal Classification:" Grid.Column="2" Grid.Row="0"/>
   <WebFront:ComboBoxTreeView PropertyName="PortalClassification" Grid.Column="2" Grid.Row="1"/>
</Grid>
```

Margin

As you see in the above example the columns are very close to each other and don't align to the other fields of the form. This can be adjusted using Margin.





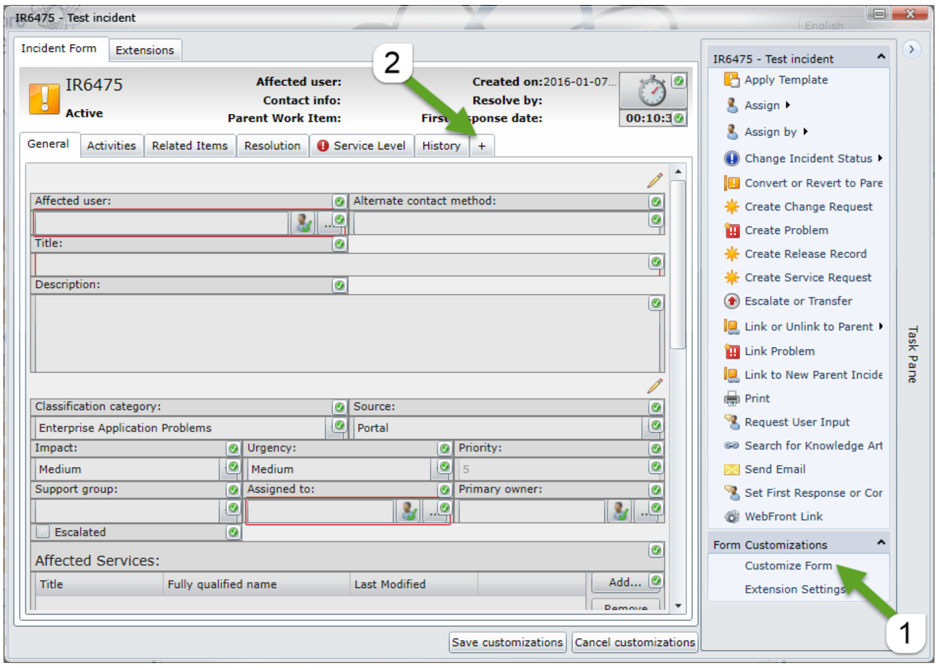
```
<Grid >
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
   </Grid.ColumnDefinitions>
   <TextBlock Text="Affected User:" Grid.Column="0" Grid.Row="0" Margin="0,0,0,0" />
   <WebFront:UserPickerWrapper Alias="AffectedUser" Grid.Column="0" Grid.Row="1" Margin="0,0,0,0"  />
   <TextBlock Text="Contact Method:" Grid.Column="1" Grid.Row="0" Margin="5,0,0,0" />
   <TextBox Text="{Binding Path=[ContactMethod].Value,Mode=TwoWay}" Grid.Column="1" Grid.Row="1" Margin="5,0,0,0" />
   <TextBlock Text="Portal Classification:" Grid.Column="2" Grid.Row="0" Margin="5,0,0,0" />
   <WebFront:ComboBoxTreeView PropertyName="PortalClassification" Grid.Column="2" Grid.Row="1" Margin="5,0,0,0"  />
</Grid>
```

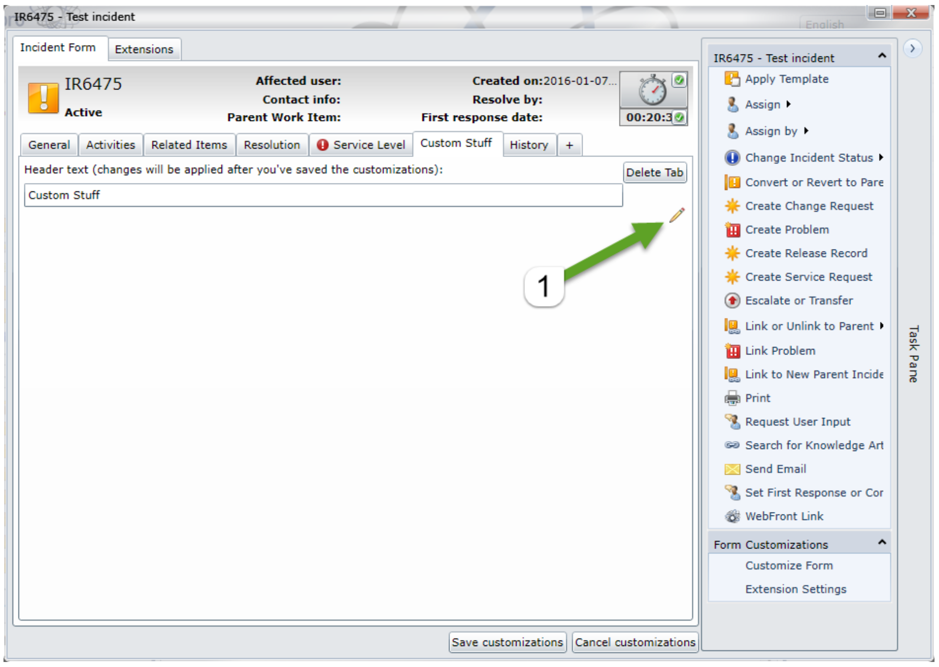
Custom Tabs

When customizing forms in WebFront you have the possibility to add custom tabs to a form. When added, the custom tabs appear on the left-hand side of the history tab in a form. Each custom tab has its own area which can be customized with XAML code to visualize data from the Service Manager data model.

Adding a Custom Tab

WebFront has a Customization Editor that can be enabled from within each WebFront form.

To launch the Customization Editor a form in WebFront	
☐	Open WebFront as a Service Manager Administrator
☐	Open the form you want to customize by simply viewing a suitable work item or configuration item associated with the form
☐	In the task menu to the right, click **Customize Form** *Arrow 1 in image below*
☐	Click the plus sign, pointed out as number 2 in the image below  IR6475 - Test incident IR6475 Active Created on: 2016-01-07... 00:10:3 Classification category: Enterprise Application Problems Source: Portal Impact: Medium Urgency: Medium Priority: 5 Incident Form Extensions Affected user: Contact info: Parent Work Item: Resolve by: First Response date: General Activities Related Items Resolution Service Level History + Alternate contact method: Title: Description: Support group: Assigned to: Primary owner: Escalated Affected Services: Title Fully qualified name Last Modified Add... Remove Save customizations Cancel customizations Apply Template Assign Assign by Change Incident Status Convert or Revert to Pare Create Change Request Create Problem Create Release Record Create Service Request Escalate or Transfer Link or Unlink to Parent Link Problem Link to New Parent Incide Print Request User Input Search for Knowledge Art Send Email Set First Response or Cor WebFront Link Form Customizations Customize Form Extension Settings Task Pane 1 2
☐	When prompted, give the tab a name and click OK

☐	To add content to your custom tab, click the pencil in the top right corner *Arrow 1 in image below* 
☐	When done, click Save customizations *Note: If you have changed the header text on a tab, previously added changes will not take effect until after closing and re-opening the form*

IR6475 - Test incident

Header text (changes will be applied after you've saved the customizations):

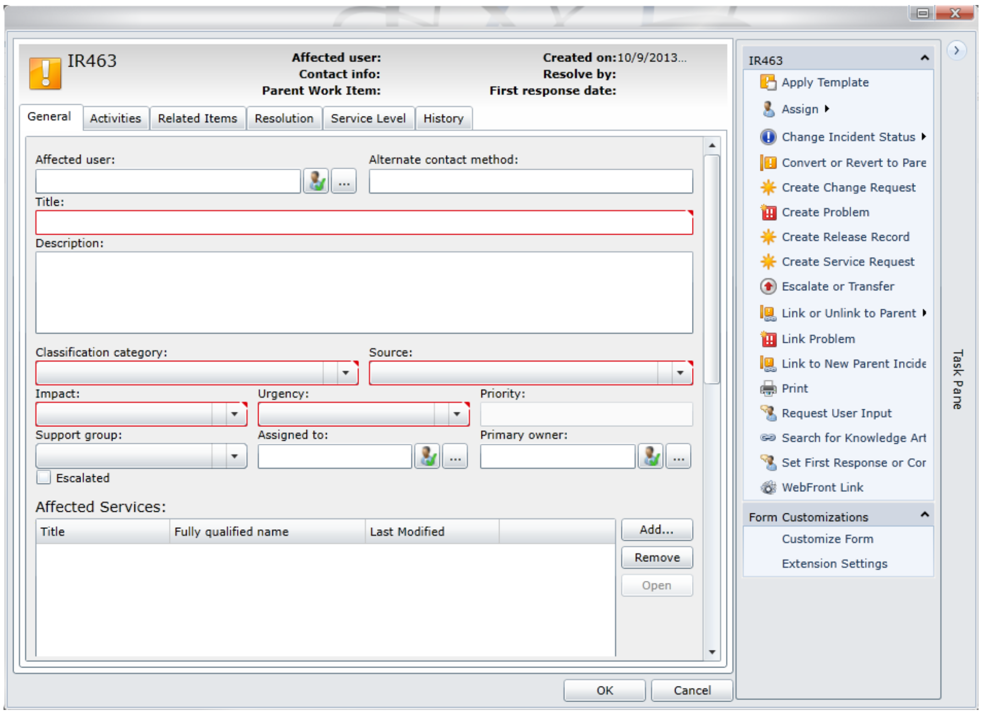
Custom Stuff

Customization Example 1

Let's go through the steps of adding a textbox bound to a custom property "CallerContactInfo" and a "label" for usability.

In this example, we will customize the form to show a TextBox control bound to a data model extension. The extension (a string property) were added using the System Center Service Manager Authoring Tool. To learn more about the System Center Service Manager Authoring Tool, see: http://technet.microsoft.com/en-us/library/hh495563.aspx

Before we start, let's take a look at the Incident Form in WebFront and how it looks out-of-the-box, this can be seen in figure 3.



IR463

Affected user:
Contact info:
Parent Work Item:

Created on:10/9/2013...
Resolve by:
First response date:

Figure 3. Incident Form.

Let's customize the top zone on the General Tab

To open the Customization Editor for the uppermost customizable zone on the General tab in the Incident Form	
☐	Open WebFront as a Service Manager Administrator
☐	Open the incident form by simply viewing any existing incident
☐	In the task menu to the right, click **Customize Form**
☐	Click the uppermost pencil ( ) on the **General** tab *Arrow 2 in figure 1*

To add a TextBox bound to the property CallerContactInfo	
☐	In the list of properties, in the upper right corner, select the property you want to bind to (in our case the extension property called CallerContactInfo) *Note: Since the CallerContactInfo property is a string property, note how the TextBlock (a read-only control) and the TextBox is tagged with a check mark since these controls are suitable to display string values*
☐	In the list of controls, select the TextBox control *Note: How the snippet is generated with pre-defined bindings to the property*
☐	Copy the code snippet *Code snippet:* *<TextBox Text="{Binding Path=[CallerContactInfo].Value,Mode=TwoWay}"/>*

☐	In the XAML Editor, replace the text **"<-- Put your code here…. >**" with the code snippet
	Result should look like below:
	<Grid>
	    <Grid.RowDefinitions>
	        <RowDefinition Height="Auto"/>
	        <RowDefinition Height="Auto"/>
	    </Grid.RowDefinitions>
	    <Grid.ColumnDefinitions>
	        <ColumnDefinition/>
	        <ColumnDefinition/>
	    </Grid.ColumnDefinitions>
	    <TextBox Text="{Binding Path=[CallerContactInfo].Value,Mode=TwoWay}"/>
	</Grid>

☐	To add a “label”, select the control called TextBlock
☐	Copy the snippet
☐	Add the snippet above the previously added snippet

☐	Change the Text attribute of the pasted TextBlock control so that it looks like below:
	
	`<TextBlock Text="Caller contact info:"/>`
	
	Result should look like below:
	`<Grid>`
	  `<Grid.RowDefinitions>`
	    `<RowDefinition Height="Auto"/>`
	    `<RowDefinition Height="Auto"/>`
	  `</Grid.RowDefinitions>`
	  `<Grid.ColumnDefinitions>`
	    `<ColumnDefinition/>`
	    `<ColumnDefinition/>`
	  `</Grid.ColumnDefinitions>`
	  `<TextBox Text="{Binding Path=[CallerContactInfo].Value,Mode=TwoWay}"/>`
	`</Grid>`

☐	To move the TextBox to the second row, add a Grid.Row attribute to the TextBox element as below:
	
	`<TextBox Text="{Binding Path=[CallerContactInfo].Value,Mode=TwoWay}"`
	`Grid.Row="1"/>`

☐	To get a preview of the customization, click Refresh in the top left corner of the Customization Editor
☐	Click to close Customization Editor
☐	To apply the customizations, click Save Customization in the bottom right corner of the form to save the changes
☐	In the Confirmation dialog, click Yes to save the customizations
☐	In the Incident Form, click “Customize Form” to hide the pencils

The customized form can be seen in figure 4

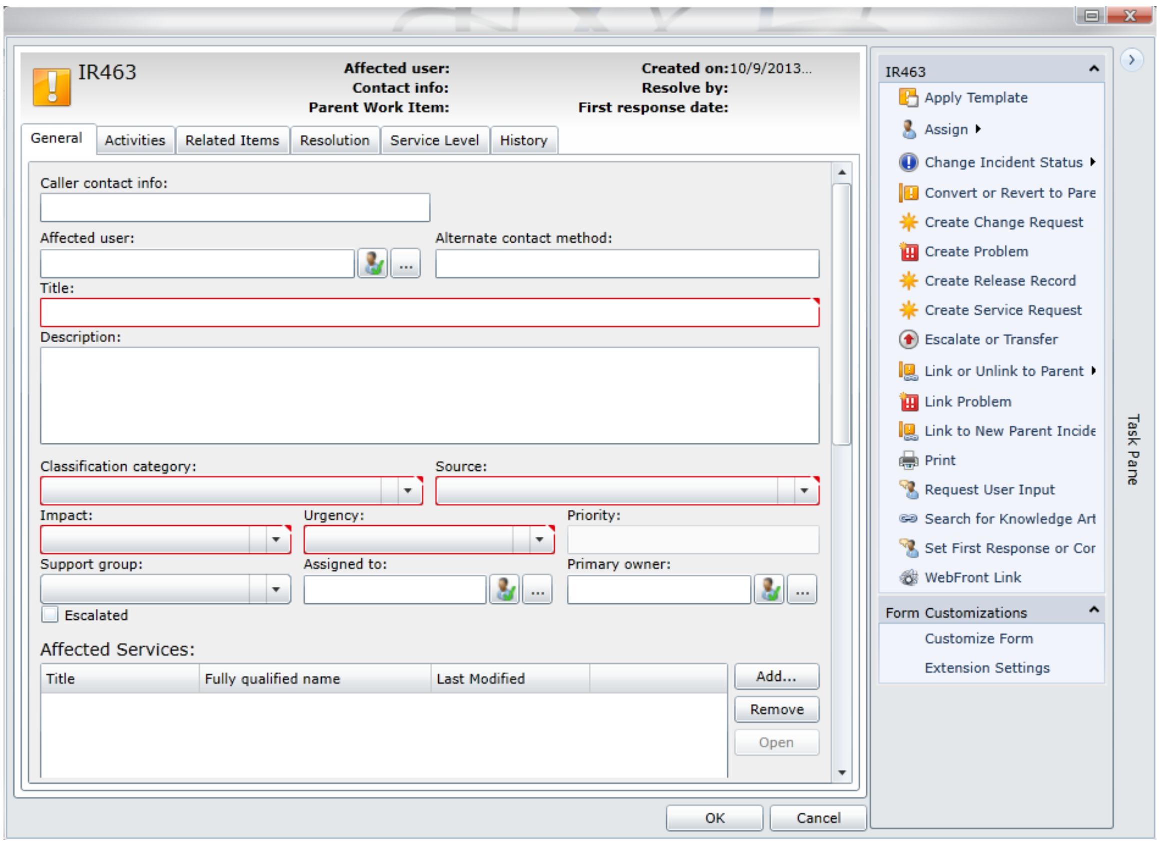


Figure 4. Incident Form with customization.

Customization Example 2

Let's go through the steps of adding a User Picker to the customizations already added in Example 1. The User Picker will be bound to a custom relationship that was created using the Microsoft Service Manager Authoring Tool. The relationship is a one-to-one relationship between the WorkItem class and the Sysem.User class.

To learn more about the System Center Service Manager Authoring Tool, see: http://technet.microsoft.com/en-us/library/hh495563.aspx

Important when binding to custom relationships (e.g. when using the User Picker)
Class extensions will be available for binding directly after importing them in Service Manager. To be able to leverage a custom relationship in your form customizations, like when using the User Picker, you need to make sure that the relationship is defined as a component in the type projection used by the form you are customizing. Since WebFront uses the same type projections for forms as the standard console this means that you only need to make sure this is the case in the standard console. The easiest way to do this is to customize the standard console form and bind a User Picker to the custom relationship since the Service Manager Authoring Tool will create a custom type projection that contains a component based on the custom relationship and associate the form with the new type projection without you having to know the first thing about type projections and how to associate these with a form.

To open the Customization Editor for the uppermost customizable zone on the General tab in the Incident Form	
☐	Start WebFront as a Service Manager Administrator
☐	Open the incident form by simply viewing any existing incident
☐	In the task menu to the right, click "Customize Form"
☐	Click the uppermost pencil (✏) on the General tab *Arrow 2 in figure 1*

To add a User Picker bound to a custom component (relationship)	
☐	In the list of components, second tab in the upper right corner, select the component you want to bind to *When the System Center Service Manager Authoring Tool takes care of creating the custom type projection the component is given an auto generated name. In our case, we had created a custom relationship called WorkItemHasCaller and the component was given the name ComponentAlias_944173e5_72f2_4db9_9e2d_5d1108a188ca.*

	Tip: To find the correct component if you have several “auto named components” is to hover the components in the list, the tooltip shown is the name of the relationship used by the component.
☐	In the list of controls select the User Picker control *Note: How the snippet is generated with pre-defined bindings to the property*
☐	Copy the code snippet *Code snippet: <WebFront:UserPickerWrapper Alias="ComponentAlias_944173e5_72f2_4db9_9e2d_5d1108a188ca"/>*
☐	In the XAML Editor, insert the code snippet below the TextBox element
☐	In the list of controls, select the TextBlock control
☐	Copy the snippet
☐	Add the snippet above the previously added snippet
☐	Change the Text attribute of the pasted TextBlock control so that it looks like below: <TextBlock Text="Caller:"/>

☐	Let’s adjust the placement, Grid.Column ,Grid.Row and Margin, attributes of the four added controls as below:
	<Grid >
	    <Grid.RowDefinitions>
	        <RowDefinition Height="Auto" />
	        <RowDefinition Height="Auto" />
	    </Grid.RowDefinitions>
	    <Grid.ColumnDefinitions>
	        <ColumnDefinition />
	        <ColumnDefinition />
	    </Grid.ColumnDefinitions>
	    <TextBlock Text="Caller contact info:" Grid.Column="1" Margin="5,0,0,0" />
	    <TextBox Text="{Binding Path=[CallerContactInfo].Value,Mode=TwoWay}" Grid.Row="1" Grid.Column="1" Margin="5,0,0,0" />
	    <TextBlock Text="Caller:" Margin="0,0,5,0" />
	    <WebFront:UserPickerWrapper Alias="ComponentAlias_944173e5_72f2_4db9_9e2d_5d1108a188ca" Grid.Row="1" Margin="0,0,5,0" />
	</Grid>

☐	To get a preview of the customization, click Refresh in the top left corner of the Customization Editor
☐	Click OK in the bottom right corner of the Customization Editor
☐	To apply the customizations, click Save Customization in the bottom right corner of the form to save the changes
☐	In the Confirmation dialog, click Yes to save the customizations
☐	In the Incident Form, click "Customize Form" to hide the pencils

The customized form can be seen in figure 5.

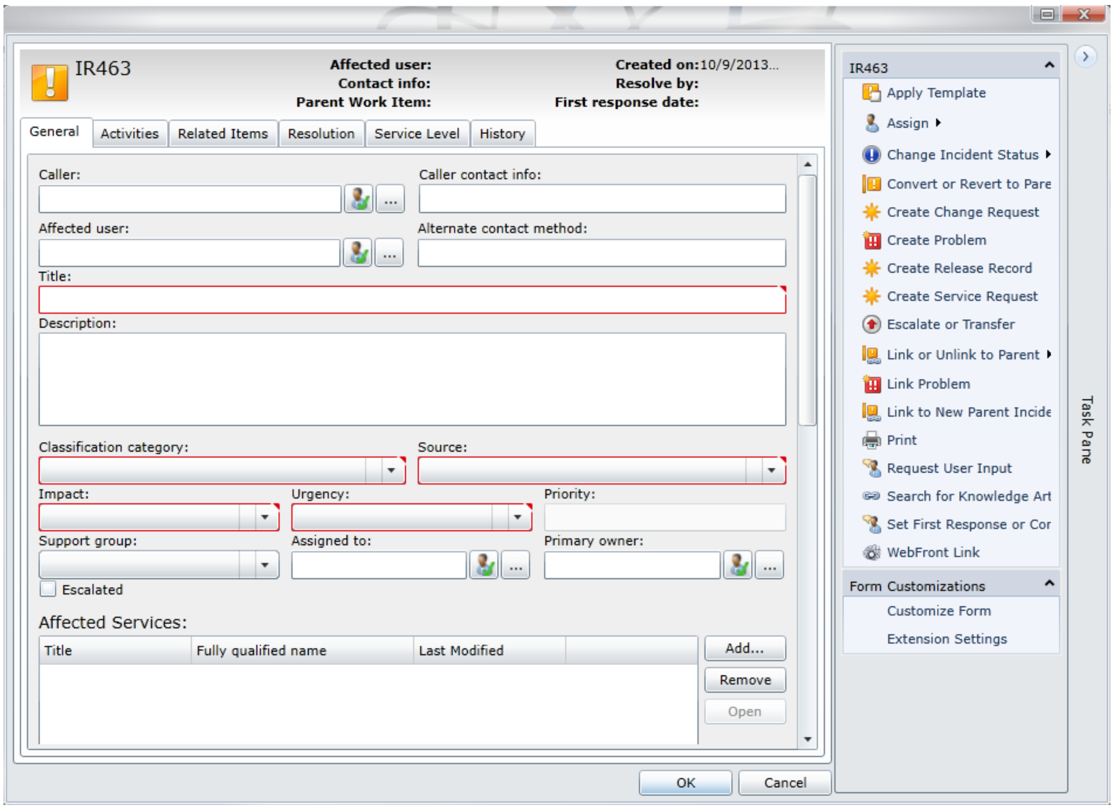
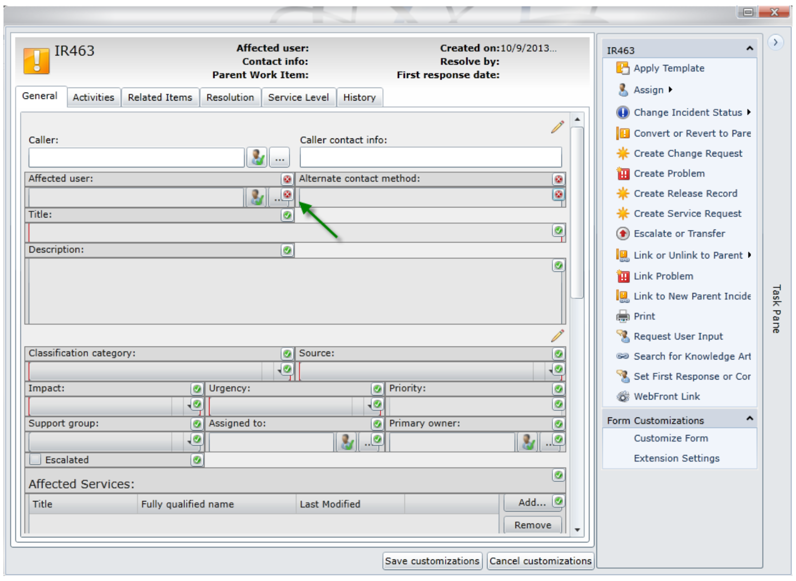
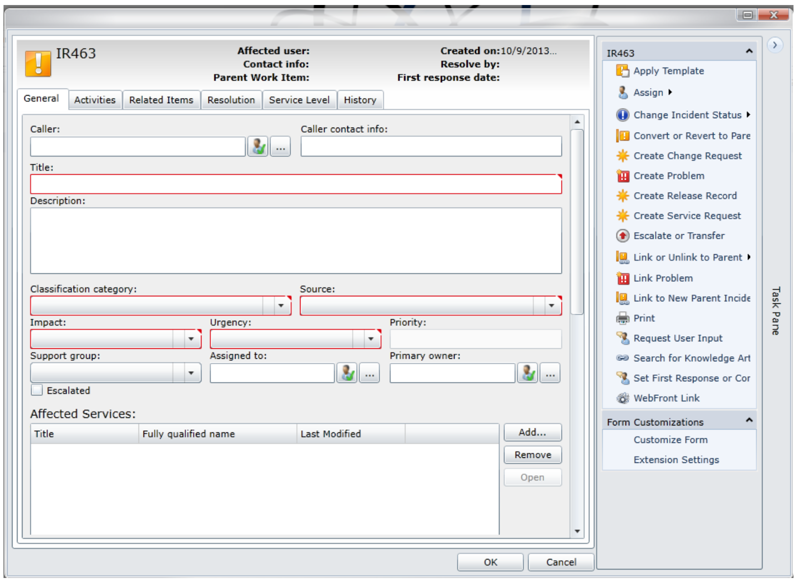

IR463
Affected user:
Contact info:
Parent Work Item:
Created on: 10/9/2013...
Resolve by:
First response date:

Figure 5. Incident Form with customization.

This scenario could be extended to replace Affected User and Alternate contact method with Caller and Caller contact info fields	
☐	Start WebFront as a Service Manager Administrator
☐	Open the incident form by simply viewing any existing incident
☐	In the task menu to the right, click "Customize Form"

☐	Click the green checkbox above the Affected User and Alternate contact fields and labels so that a red stop sign is shown 
☐	Click **Save customizations**
☐	See result below where we have replaced controls 

Supported Controls

In this version of WebFront for Service Manager we support the following control types in terms of providing code snippets automatically:

- ComboBox
- DateTimePicker
- TextBlock
- TextBox
- UserPicker
- SingleInstancePicker
- Checkbox

NOTE: You are still able to use all native Silverlight controls, it is just that WebFront will not generate the code snippet for you.

Storage of Customizations

Form customizations made in WebFront are stored in the Service Manager database.

Custom Branding

It is possible to replace the top left logo and center logo. For more information about this see the Deployment Guide.