

Oracle® Agile Product Lifecycle Management for Process

Sidebar Extensibility Guide

Feature Pack 4.0

E63497-01

April 2015

ORACLE®

Copyrights and Trademarks

Agile Product Lifecycle Management for Process

Copyright © 1995, 2015 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle

Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

| | |
|---|-----------|
| PREFACE..... | 5 |
| Audience | 5 |
| Variability of Installations | 5 |
| Documentation Accessibility..... | 5 |
| Access to Oracle Support | 5 |
| Software Availability | 5 |
| CHAPTER 1—OVERVIEW | 6 |
| CHAPTER 2—SIDEBAR USER INTERFACE | 6 |
| Expanded Sidebar | 6 |
| Minimized Sidebar | 8 |
| Sidebar Content | 8 |
| CHAPTER 3—SIDEBAR CONFIGURATION | 9 |
| Common..... | 9 |
| Common Attributes | 9 |
| Display..... | 9 |
| Display Attributes..... | 9 |
| Panel | 10 |
| Panel Attributes | 10 |
| Widget Attributes | 11 |
| DynamicPanel | 12 |
| DynamicPanel Attributes | 12 |
| CHAPTER 4—SIDEBAR JAVASCRIPT APIS..... | 14 |
| Sidebar functions | 14 |
| Panel functions | 16 |
| Widget functions..... | 16 |
| Sidebar events..... | 21 |
| CHAPTER 5—A BI PUBLISHER REPORT EXAMPLE..... | 22 |

Preface

Audience

This guide is intended for client programmers involved with integrating Oracle Agile Product Lifecycle Management for Process. Information about using Oracle Agile PLM for Process resides in application-specific user guides. Information about administering Oracle Agile PLM for Process resides in the *Oracle Agile Product Lifecycle Management for Process Administrator User Guide*.

Variability of Installations

Descriptions and illustrations of the Agile PLM for Process user interface included in this manual may not match your installation. The user interface of Agile PLM for Process applications and the features included can vary greatly depending on such variables as:

- Which applications your organization has purchased and installed
- Configuration settings that may turn features off or on
- Customization specific to your organization
- Security settings as they apply to the system and your user account

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Software Availability

Oracle Software Delivery Cloud (OSDC) provides the latest copy of the core software. Note the core software does not include all patches and hot fixes. Access OSDC at:

<http://edelivery.oracle.com>

Chapter 1—Overview

The PLM for Process user interface provides an extensible area for customers to include their own UI components, which are displayed alongside application pages. This Sidebar can host various components that can be launched by the user as needed. These components can display useful contextual information, such as a specific BI report based on the current specification, a historical event listing of object, and out-of-the-box Hierarchy Navigator, and more.

The PLM for Process Sidebar also provides helpful features, including easily selecting placement of the sidebar, launching a menu of components to display, and more. A set of helpful JavaScript functions are available to the sidebar components that can act on the sidebar's appearance and behavior.

This document describes the features of the PLM for Process Sidebar, how it can be configured, and the JavaScript library available to the components. An example of a BI Publisher report sidebar component is also presented.

Chapter 2—Sidebar User Interface

The Sidebar content is displayed inside *panels*. One panel can be displayed at a time. Each panel contains one or more components (called *widgets* in the configuration file) which are displayed at the same time.

The sidebar is hidden by default; to show it, a navigation extension can be added using an onclick event which simply calls the JavaScript to show the sidebar. For instance, the following JavaScript call will open the sidebar in the left content area: `NavigatorSidebar.showLeftSidebar();`

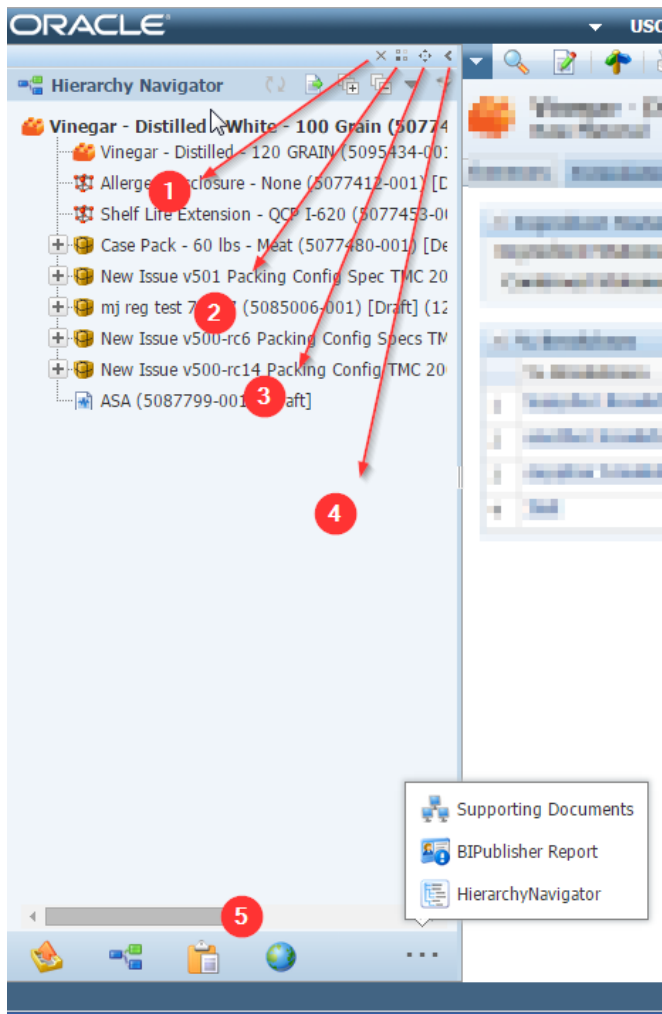
The following navigation extension example, in the SiteMap-Extensions.xml file, shows how you can add a button to GSM specs screens which will open the sidebar:

```
<MenuItem ID="GSMNavSpec" configChildKey="ID">
  <MenuItem ID="toolbuttons" configChildKey="ID">
    <MenuItem MenuType="icon" DisplayText="Show Sidebar" CssClass="lblHierarchyNavigator"
SortOrder="179" ClientSideCommand="NavigatorSidebar.showSidebar();" ></MenuItem>
  ...
</MenuItem>
```

Expanded Sidebar

The expanded Sidebar user interface allows you to view a sidebar panel and its component(s), select another available panel to view, and manage the list of panels. In addition, the Sidebar can be minimized, closed, or moved/dragged to the opposite side of the screen.

The following screenshot illustrates the expanded sidebar and the features available:



1. Close Sidebar

Closes the sidebar and any active panels.

2. Panel Manager

Launches the Panel Manager pop-up for users to select which panels they want to have available in their panel listing. See [Panel functions](#) for more information.


3. Drag and Drop

The Sidebar can be placed on the left or right side of the screen. Use the Drag and Drop icon to drag the panel to one side or the other.

4. Minimize

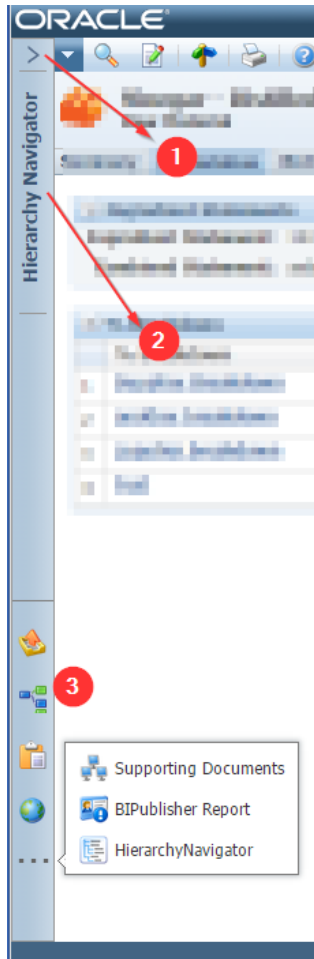
You can minimize the Sidebar by selecting this action. See [Minimized Sidebar](#) for more information about the minimized panel.

5. Available Panels

Available panels are displayed as icons in the bottom of the Sidebar. A limited number of panel icons will be displayed; click the ellipses button  to see the remaining Panels available.

Minimized Sidebar

A minimized Sidebar provides quick access to the currently active panel, as well as links to the other available panels.



1. Expand Sidebar

Opens the Sidebar, and displays any currently active panel.

2. Active Panel Name

The name of the panel that is currently active.

3. Available Panels

Available panels are displayed as icons in the bottom of the Sidebar. A limited number of panel icons will be displayed; click the ellipses button



to see the remaining Panels available.

Sidebar Content

The contents and basic properties of the Sidebar are specified through the Sidebar configuration file. Each component is rendered based on its configuration, which includes icons, titles, control over visibility, and the way it should render its content. The following section describes the configuration details.

Chapter 3—Sidebar Configuration

The Sidebar is configured using the config\Extensions\SidebarConfig.xml file.

Common

The common node defines common settings for the Sidebar.

```
<Common>
  <EnableSidebar>true</EnableSidebar>
</Common>
```

Common Attributes

| Attribute | Description |
|----------------------|--|
| EnableSidebar | Turns the sidebar tool on or off. The value of the EnableSidebar should be true or false |

Display

The display node defines some common UI settings for the Sidebar.

```
<Display>
  <SidebarPosition>right</SidebarPosition>
  <DefaultWidth>320</DefaultWidth>
  <MinWidth>320</MinWidth>
  <MaxWidth>1000</MaxWidth>
</Display>
```

Display Attributes

| Attribute | Description |
|------------------------|--|
| SidebarPosition | Defines the default position of the Sidebar on the screen, until the user overrides it by dragging the sidebar to the other side. Valid values are <i>left</i> or <i>right</i> |
| DefaultWidth | Defines the default width, in pixels, of the Sidebar, until the user overrides it by dragging the panel wider. |
| MinWidth | Defines the minimum width, in pixels, of the Sidebar. |
| MaxWidth | Defines the maximum width, in pixels, of the Sidebar |

Panel

The Panel node defines the panels that are available. Each panel contains one or more components (called widgets in the configuration file) which are displayed at the same time

There are two kinds of panels in the configuration:

- Panel – A panel defines its components/widgets in the configuration file.
- Dynamic Panel – A DynamicPanel will load its configuration from an external source, via a URL, rather than from the configuration file. This allows for more widgets that are dynamically controlled. To learn more about Dynamic Panels, see the next section.

Panels contain one or more widgets, which account for the actual panel content. Although a panel can contain multiple widgets, the space available in the Sidebar UI limits the actual amount of information that can be displayed. Therefore, the number of widgets in a single panel will depend on the contents of each widget and should be limited to no more than three.

```
<Panel ID="PANEL_ID"
      TranslationID="TRANSLATION_ID"
      Title="TITLE"
      Icon="CSS_CLASS"
      DefaultShow="true">

  <Widget...></Widget>
  <Widget...></Widget>

</Panel>
```

Panel Attributes

| Attribute | Description |
|----------------------|---|
| ID | A unique identifier of the panel. |
| TranslationID | The title of the panel will be displayed for the user's current language based on this translation identifier. Translations must be added to the <code>commonXLAExtensionCacheItem</code> table, using the <code>fkParent ID</code> from <code>commonXLAExtensionCacheItem</code> table (<code>select pkid from commonXLAExtensionCache where Name like '%NavigatorSidebar%';</code>) |
| Title | The title of the panel to display when not using translations. Note: if you set both TranslationID and Title, then TranslationID will be used for the title. |
| Icon | The css class that will used in the Sidebar for this panel's icon. For instance: <pre><Panel ... Icon="cssx-demo-bookmark" ... ></pre> And the related css <pre>.cssx-demo-bookmark { background-image: url(/gsm/images/navigatorSidebar/demo/hierarchy.png); }</pre> |

| Attribute | Description |
|--------------------|--|
| DefaultShow | Specifies whether the panel should be available when user opens the Sidebar. The value should be true or false. If the DefaultShow attribute is not specified, the value will be false. If false, the user can still select the panel through the Extension Manager. |

The Widget configuration determines which components are loaded inside a panel. Each widget has its own icon and title, which can match the panel's values, but should differ when there are multiple widgets inside a panel.

```
<Widget
  ID="WIDGET_ID"
  Title="WIDGET_TITLE"
  TranslationID="TRANSLATION_ID"
  Icon="CSS_CLASS"
  UserControlURL="URL"
  Height="300"
  EnabledObjectType="TYPE_IDS"
  EnabledFactory="FACTORY_CLASS_PATH">
</Widget>
```

Widget Attributes

| Attribute | Description |
|--------------------------|--|
| ID | A unique identifier of the widget. |
| TranslationID | The title of the widget will be displayed for the user's current language based on this translation identifier. Translations must be added to the <code>commonXLAExtensionCacheItem</code> table, using the <code>fkParent</code> ID from <code>commonXLAExtensionCacheItem</code> table (<code>select pkid from commonXLAExtensionCache where Name like '%NavigatorSidebar%';</code>) |
| Title | The title of the widget to display when not using translations. Note: if you set both TranslationID and Title, then TranslationID will be used for the title. |
| Icon | The css class that will be used in the Sidebar for this widget's icon. For instance: <pre><Widget ... Icon="cssx-demo-bookmark" ... ></pre> And the related css <pre>.cssx-demo-bookmark { background-image: url(/gsm/images/navigatorSidebar/demo/hierarchy.png); }</pre> |
| Height | The height of the widget, in pixels. Generally should only be specified if needed; otherwise, the container panel will handle the height automatically. |
| EnabledObjectType | Comma separated list of object types that this widget will be enabled for. For instance, "1004,2147" will enable this widget for Material and Trade Specifications. See the Extensibility Guide Appendix for details about Object Types. |

| Attribute | Description |
|---|---|
| EnabledFactory | <p>ObjectLoaderURL for the class that controls if the widget should be enabled in current context. Ignored if EnabledObjectType is used.</p> <p>For example:</p> <pre>Class:Xeno.Web.UI.Common.NavigatorSidebar.Demo.HasSpecEnabledFactory,WebCommon</pre> <p>To implement this class you must implement the Xeno.Web.UI.Common.NavigatorSidebar.ISidebarEnabled interface from the WebCommon.dll assembly, and implement the following method:</p> <pre>public bool CheckEnabled(string currentObjectPKID)</pre> <p>See the Extensibility Guide Appendix for details about ObjectLoaderURL syntax.</p> |
| The displayed content of the widget are specified through <i>one</i> of the following attributes: | |
| RemoteURL | <p>Loads the widget content from the specified URL, such as a BI report. For example: "http://www.oracle.com?get={ObjectPkid}"</p> <p>Supported Parameters: {ObjectPkid} and {ObjectType}</p> |
| UserControlURL | <p>Load the widget content from a .NET user control.</p> <p>For example:</p> <pre>~/WebCommon/NavigatorSidebar/HelloWorld.ascx.</pre> |

DynamicPanel

The DynamicPanel node defines panels that load widgets dynamically. A URL returns a JSON object representing a panel and one or more widgets.

```
<DynamicPanel ID="MRU" ConfigUrl="http://example.com/panelDefinition.js" DefaultShow="true" />
```

DynamicPanel Attributes

| Attribute | Description |
|--------------------|---|
| ID | A unique identifier of the panel. |
| DefaultShow | <p>Specifies whether the panel should be available when user opens the Sidebar. The value should be true or false. If the DefaultShow attribute is not specified, the value will be false.</p> <p>If false, the user can still select the panel through the Extension Manager.</p> |
| ConfigUrl | <p>Url that returns a panel definition via a JSONP format data. The JSON object has the most of the same properties and behaviors as the Panel and Widget configuration settings defined above. See below for details.</p> <p>For example:</p> <pre>NavigatorSidebarPanel ({ "ID": "RemoteWidget", "Icon": "{PanelIconUrl}", "Title": "{PanelTitle", DynamicPanel doesn't support TranslationID property", "Widgets": [{ "ID": "{widgetId}", "Title": "{widgetTitle}",</pre> |

| Attribute | Description |
|-----------|---|
| | <pre> "RemoteURL": "{RemoteUrl, this priority is higher than Content. If user sets RemoteURL and Content at the same time, Content property will be ignored}", "Content": "{HTMLContent}", "Icon": "{WidgetIcon}", "Height": {widgetHeight, this property can be null}} }}</pre> <p>Note:</p> <ul style="list-style-type: none"> ▪ Supported parameters for ConfigUrl: {ObjectPkid} and {ObjectType}. For example: <ul style="list-style-type: none"> ○ <code><DynamicPanel ID="RemoteWidget" ConfigUrl="http://example.com/loadpanel.aspx?pkid={ObjectPkid}&type={ObjectType}"></DynamicPanel></code> ▪ Panel config must wrapped by NavigatorSidebarPanel() ▪ Widget config does not support EnabledFactory attribute, but does support EnabledObjectType. |

Chapter 4—Sidebar JavaScript APIs

A set of JavaScript APIs are available that provide functions to manipulate the Sidebar, along with panels and widgets. Custom widgets can access these functions via the JavaScript object variable

NavigatorSidebar. For example, the `NavigatorSidebar.closeSidebar()` function will close the sidebar.

Sidebar functions

The following functions are available to widgets.

| Sidebar Javascript API List | |
|-----------------------------|--|
| showSidebar() | Parameter(s): None Returns: n/a |
| | Description: Show the Sidebar if it is closed. The position of the sidebar will be based on user preference (last position that user opened). |
| showLeftSidebar() | Parameter(s): None Returns: n/a |
| | Description: Show the Sidebar on the left side of the page |
| showRightSidebar() | Parameter(s): None Returns: n/a |
| | Description: Show the Sidebar on the right side of the page |
| expandSidebar() | Parameter(s): None Returns: n/a |
| | Description: Expand Sidebar if it is minimized |
| collapseSidebar() | Parameter(s): None Returns: n/a |
| | Description: Collapse/minimize the Sidebar |
| closeSidebar() | Parameter(s): None Returns: n/a |
| | Description: Close the Sidebar completely. |
| addWidget(panelId,widget) | Parameter(s): panelId : panel id that hosts widget widget : JSON object. e.g. <code>{"ID":id,"Title":title,"Content":widgetContent,"Height":widgetHeight,"Icon":iconclass}</code> Note: widgetContent could be YUI object or JQuery object or pure html strings |
| | Returns: Widget object. Refer to Widget API List for more details |
| | Description: Add widget into specific panel |

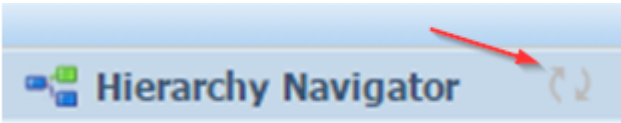
| | |
|------------------------------------|--|
| getWidget(panelId,widgetId) | <p>Parameter(s):</p> <p>panelId: id of the panel that hosts widget</p> <p>widgetId: the unique id of the widget to retrieve</p> <p>Returns: Widget object. Refer to Widget API List for more details</p> <p>Description:</p> <p>Get widget by panelId and widgetId</p> |
| addWidgetPanel(id,title,iconClass) | <p>Parameter(s):</p> <p>id: panel id</p> <p>title: panel title</p> <p>IconClass: panel icon css class</p> <p>Returns: Panel object. Refer to Panel API List for more details</p> <p>Description: Add panel to Sidebar</p> |
| getWidgetPanel(panelId) | <p>Parameter(s):</p> <p>panelId: the unique id of the pane to retrieve</p> <p>Returns: Panel object. Refer to Panel API List for more details</p> <p>Description: Get specific panel</p> |
| getAllPanelId() | <p>Parameter(s): None</p> <p>Returns: List of all panel ids</p> <p>Description: Get all panel id list that loaded in Sidebar</p> |
| on(eventName,callbackFunc) | <p>Parameter:</p> <p>eventName: event name that you want to subscribe to</p> <p>callbackFunc: callback function when subscribed event fires</p> <p>Refer to Sidebar Event List for available events</p> <p>Returns: n/a</p> |
| showPanel(panelId) | <p>Parameter(s):</p> <p>panelId: id of the panel to display</p> <p>Returns: n/a</p> <p>Description:</p> <p>Show specific panel. You can use this api to switch between opened panels.</p> |

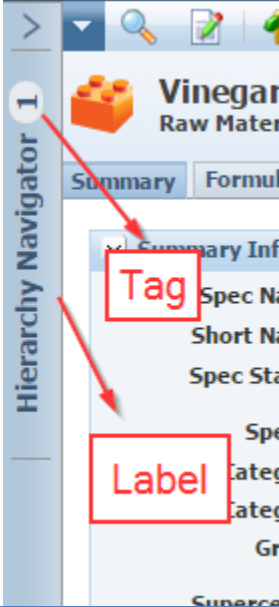
Panel functions

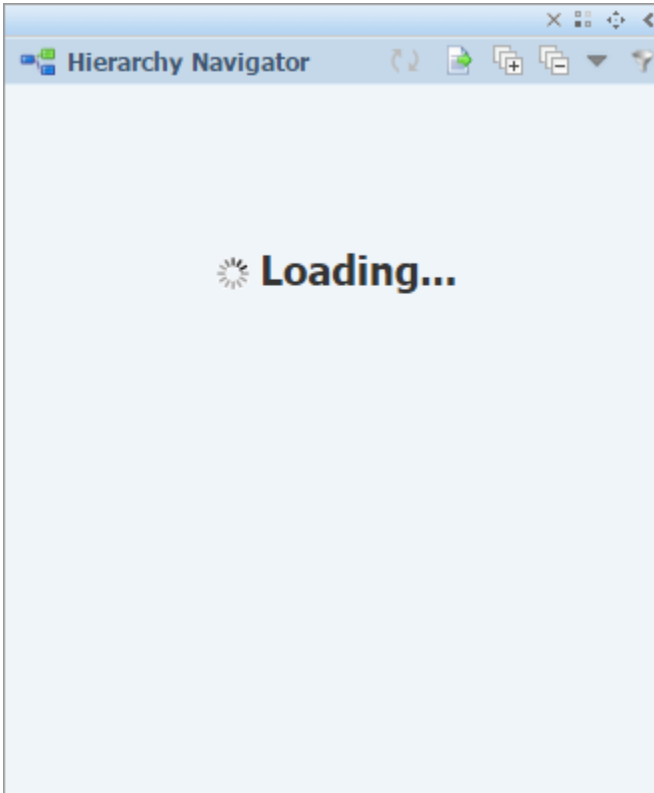

| Sidebar Panel API List | |
|------------------------|------------------------------------|
| show () | Parameter(s): None Returns: n/a |
| | Description: Show current panel. |
| remove() | Parameter(s): None Returns: n/a |
| | Description: Remove current panel. |


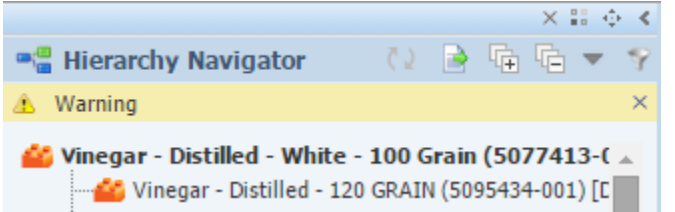
Widget functions

| Sidebar Widget API List | |
|---------------------------------|---|
| remove () | Parameter(s): None Returns: n/a |
| | Description: Remove current widget. Example: <pre>var widget = getWidget(<panelId>,<widgetId>); widget.remove();</pre> |
| updateContent (content) | Parameter(s): content: content could be YUI object or JQuery object or pure html strings Returns: n/a |
| | Description: Update content of the widget Example: <pre>var widget = getWidget(panelId,widgetId); widget. updateContent ("test");</pre> |
| addButton(button,clickCallback) | Parameter(s): button: json object. e.g. {"ID":id,"Title":title,"Content":widgetContent} Content could be either YUI object or JQuery object or pure html strings. clickCallback: callback function after clicked the button |

| | |
|------------------------|--|
| | Returns: n/a |
| | Description: Add button to the right of the widget title, like the following screenshot  |
| removeButton(buttonId) | Parameter(s): buttonId : id of the button that to remove Returns: n/a |
| | Description: Remove specific widget button |
| AddMenu(menuObject) | Parameter(s): menuObject : json object, for example: <pre>{ "ID": "menuId", "Title": "MenuTitle", "Icon": "", "Menu": [{ "ID": "subMenuId", "Title": "title", "Callback": function }, { "ID": " subMenuId2", "Title": " title 2", "Callback": function }] }</pre> |
| | Returns: n/a Description: like the addButton function, this will add a structured menu button to the right of the widget title. |
| removeMenu(menuId) | Parameter(s): menuId : id of the menu that you want to remove Returns: n/a |
| | Description: Remove specific widget menu |
| setLabel (txt) | Parameter(s): txt : label text Returns: n/a |

| | |
|---------------|--|
| | <p>Description: Set widget label for the minimized panel</p>  |
| setTag(txt) | <p>Parameter(s): txt: tag text (typically a number)</p> <p>Returns: n/a</p> <p>Description: Set widget tag, next to the label, for the minimized panel</p> |
| clearTag() | <p>Parameter(s): None</p> <p>Returns: n/a</p> <p>Description: clear widget tag of a minimized panel</p> |
| showLoading() | <p>Parameter(s): None</p> <p>Returns: n/a</p> |

| | |
|---|--|
| | <p>Description: Show loading mask for current widget. For example:</p>  |
| hideLoading() | <p>Parameter(s): None Returns: n/a</p> <p>Description: Hide loading mask for current widget</p> |
| showErrorMsg(msgObject,autoDismissTime) | <p>Parameter(s): msgObject: could be YUI object or JQuery object or pure html strings autoDismissTime: (optional) dismiss timeout (millisecond).</p> <p>Returns: n/a</p> <p>Description: Show an error message at the top of the widget. For example:</p>  <p>Example:</p> <pre>var widget = getWidget(panelId,widgetId); widget.showErrorMsg ('error msg');</pre> |

| | |
|---|--|
| showSuccessfulMsg (msgObject,autoDismissTime) | <p>Parameter(s): msgObject: could be YUI object or JQuery object or pure html strings autoDismissTime: (Optional) dismiss timeout (millisecond).</p> <p>Returns: n/a</p> <p>Description: Show a successful message at the top of the widget. For example:</p>  <p>Example:</p> <pre>var widget = getWidget(panelId,widgetId); widget.showSuccessfulMsg('it works');</pre> |
| showWarningMsg (msgObject,autoDismissTime) | <p>Parameter(s): msgObject: could be YUI object or JQuery object or pure html strings autoDismissTime: (Optional) dismiss timeout (millisecond).</p> <p>Returns: n/a</p> <p>Description: Show a warning message at the top of the widget. For example:</p>  <p>Example:</p> <pre>var widget = getWidget(panelId,widgetId); widget.showWarningMsg('Warning');</pre> |
| hideMsg() | <p>Parameter(s): None</p> <p>Returns: n/a</p> <p>Description: Hide widget messages that are displayed at the top of the widget</p> |

Sidebar events

| Sidebar Event List | |
|---------------------|--|
| ready | Triggered when panel has been rendered into DOM |
| afterHideSidebar | Triggered after hide panel |
| afterShowSidebar | Triggered when panel showed from minimized status |
| startResizeSidebar | Triggered when starting to resize panel |
| afterResizeSidebar | Triggered after resized panel |
| beforeCloseSidebar | Triggered before close panel |
| afterCloseSidebar | Triggered after close panel |
| afterAddPanel | <p>Triggered after adding new panel.</p> <p>Callback function parameters: (event,panelId)</p> <p>Example: <code>NavigatorSidebar.on("afterAddPanel",function(event,panelId){});</code></p> |
| afterAddPanelWidget | <p>Triggered after adding new widget.</p> <p>Callback function parameters: (event,panelId,widgetId)</p> <p>Example: <code>NavigatorSidebar.on("afterAddPanelWidget",function(event,panelId,widgetId){});</code></p> |
| afterShowPanel | <p>Triggered after show specific panel</p> <p>Callback function parameters: (event,panelId)</p> <p>Example: <code>NavigatorSidebar.on("afterShowPanel",function(event,panelId){});</code></p> |

Chapter 5—A BI Publisher Report Example

In this chapter, we will take BI Publisher Report widget as an example and show you how to add it.

Step 1: Add config in config\Extensions\SidebarConfig.xml

In order to add BI Publisher Report widget, we need to add a new panel that hosts this widget.

Under the Panels node, we add following config:

```
<Panel ID="BIPublisher" Icon="cssx-demo-report" Title="BIPublisher Report">
    <Widget ID="Report1"
Title="BIPublisher Report"
Icon="cssx-demo-report"
EnabledFactory="Class:Xeno.Web.UI.Common.NavigatorSidebar.Demo.HasSpecEnabledFactory,WebCommon
"
configReplace="file:%CONFIG_HOME%\environmentvariables.config"
RemoteURL="@@VAR:PLM4P.ReportServer.URL@@/xmlpserver/PLMforProcess_Reference/GSM/Reports/Gener
al/Supporting%20Document%20Review.xdo?_xpf=&_xpt=0&_xdo=%2FPLMforProcess_Reference%2FG
SM%2FReports%2FGeneral%2FSupporting%20Document%20Review.xdo&_xmode=4&_xdo%3A_xdo%3A_params
DisplayLevel_div_input=Hierarchy&_paramsDisplayLevel=Hierarchy&_xdo%3A_xdo%3A_paramsSp
ecPKID_div_input=&_paramsSpecPKID={ObjectPkid}&_xt=Supporting%20Documents%20Review&
_xf=analyze&_xana=view"
></Widget>
</Panel>
```

- You should also define “cssx-demo-report” in Web\css\NavigatorSidebar.css, for example:

```
.cssx-demo-report
{
    background-image: url(/gsm/images/navigatorSidebar/demo/report.png);
}
```

- We also make a
“Class:Xeno.Web.UI.Common.NavigatorSidebar.Demo.HasSpecEnabledFactory,WebCommon”
for EnabledFactory property, which allows this widget only enabled when current page has spec.

```
using System;
using Xeno.Prodika.Application;
using Xeno.Prodika.SCRM.Service;
using Xeno.Prodika.Services;

namespace Xeno.Web.UI.Common.NavigatorSidebar.Demo
```


```

{
    public class HasSpecEnabledFactory : ISidebarEnabled
    {
        public ISpecificationService SpecificationService
        {
            get
            {
                return AppPlatformHelper.ServiceManager[typeof(ISpecificationService).FullName]
as ISpecificationService;
            }
        }
        public bool CheckEnabled(SidebarEnabledContext context)
        {
            try
            {
                return SpecificationService.Current != null;
            }
            catch (Exception e)
            {
                return false;
            }
        }
    }
}

```

- For the `RemoteURL`, you need to replace “&” with “amp;”, and use {ObjectPkid} as the placeholder for current spec pkid.

Step 2: Restart IIS

Restart IIS to make above config work and open Sidebar, you will see a new icon which represents the BI Publisher. Click that icon you should be able to see the following screen (based on your report, it may be different with this one). If you can't see the BI publisher icon in the Sidebar, please make sure open extension manager icon  and add BI publisher panel to your panel.

